Scientific Research Publishing

Journal of Signal and Information Processing, 2021, 12, 1-41
https://www.scirp.org/journal/jsip
ISSN Online: 2159-4481
ISSN Print: 2159-4465

A New Approach for the DFT NIST Test Applicable for Non-Stationary Input Sequences

Yehonatan Avraham, Monika Pinchas

Department of Electrical and Electronic Engineering, Ariel University, Ariel, Israel
Email: Yehonatan85@gmail.com, monika.pinchas@gmail.com, monikap@ariel.ac.il

How to cite this paper: Avraham, Y. and Pinchas, M. (2021) A New Approach for the DFT NIST Test Applicable for Non-Stationary Input Sequences. *Journal of Signal and Information Processing*, **12**, 1-41.
https://doi.org/10.4236/jsip.2021.121001

Received: November 1, 2020
Accepted: February 23, 2021
Published: February 26, 2021

Abstract

The National Institute of Standards and Technology (NIST) document is a list of fifteen tests for estimating the probability of signal randomness degree. Test number six in the NIST document is the Discrete Fourier Transform (DFT) test suitable for stationary incoming sequences. But, for cases where the input sequence is not stationary, the DFT test provides inaccurate results. For these cases, test number seven and eight (the Non-overlapping Template Matching Test and the Overlapping Template Matching Test) of the NIST document were designed to classify those non-stationary sequences. But, even with test number seven and eight of the NIST document, the results are not always accurate. Thus, the NIST test does not give a proper answer for the non-stationary input sequence case. In this paper, we offer a new algorithm or test, which may replace the NIST tests number six, seven and eight. The proposed test is applicable also for non-stationary sequences and supplies more accurate results than the existing tests (NIST tests number six, seven and eight), for non-stationary sequences. The new proposed test is based on the Wigner function and on the Generalized Gaussian Distribution (GGD). In addition, this new proposed algorithm alarms and indicates on suspicious places of cyclic sections in the tested sequence. Thus, it gives us the option to repair or to remove the suspicious places of cyclic sections (this part is beyond the scope of this paper), so that after that, the repaired or the shortened sequence (original sequence with removed sections) will result as a sequence with high probability of random degree.

Keywords

Wigner Distribution, Shape Parameter, Generalized Gaussian Distribution, Random Number Generator, True Random Number Generator, Pseudo Random Number Generator

1. Introduction

In this paper, the problem of estimating the probability of signal randomness degree is addressed, which is widespread used in cryptography applications [1] [2]. The key component of the encryption process is the Random Number Generator (RNG) [3] [4], that is a physical apparatus or a software algorithm, that generates a sequence of signs (numbers) without any visible deterministic pattern or order. A True Random Number Generator (TRNG) is an apparatus that is based on a non-deterministic entropy source while a Pseudo Random Number Generator (PRNG) is based on software or hardware [5]. The quality of an RNG is not always perfect due to the fact that physical generators (TRNG) are sensitive to external influences [6] such as temperature [7], power supply, obsolescence, may suffer from a slow and persistent failure that is not noticeable or may suffer from a sharp decline in quality [8]. In addition, generators based on software (PRNG) are defective since they are based on deterministic algorithms that generate the numbers by some formula and some initial value [9] [10].

The series obtained from the RNG should be checked if this is not a lacking random series [11]. There is a great variety of statistical tests for checking the quality of a random generator [12] [13] [14]. These tests are not meant to prove mathematically that the sequence that the generator produces is random, but rather to find out if there are any vulnerabilities in the generator that may indicate poor randomness [15]. The tests on the input sequence (a sequence of bits of ones and zeros), determine whether the tested sequence contains certain predictable characteristics in a truly random sequence. Such a test result is not deterministic but probabilistic. For example, a truly random sequence should contain zero and one bits approximately equally [14]. If the tested sequence does not meet this requirement, the generator that created the sequence fails experimentally. Conversely, if the one-bit number and zero equals a certain predetermined divergence rate, it can be said that the random sequence was "accepted". The term was "accepted" means that the claim of randomness is not rejected [16]. Even if the sequence successfully undergoes a given statistical test, it is not an absolute proof of its randomness but more than a probabilistic confirmation [17].

The NIST tests [18] are very popular [19] [20]. However, these tests (NIST) have several problems [21] [22] [23] [24] and as stated in [25], the NIST tests [18] require that the behavior of the tested sequence shall be stationary. But, according to [25], it is not rare for entropy sources (TRNGs) to generate time-varying data, since entropy sources usually depend on physical phenomena (e.g. thermal noise) and some of them are fragile and sensitive to external factors (e.g. temperature), which means the distribution of their outputs or the dependency among the outputs may change with external factors. In addition, much software based TRNGs use the computer's workload as their entropy sources, which are also variable [25].

The NIST tests [18] contain fifteen tests, based on probabilistic methods that examine the statistical independency between the bits in the tested input sequence. The result is a probability value for the "degree of randomness". This value is called "P value" [18].

In this paper, we deal with non-stationary sequences for which test number six (the DFT test from the NIST document [18]) is not suitable. Although the non-overlapping template matching test and the overlapping template matching test (test number seven and eight of the NIST document [18]) are designed to compensated for the inability of the DFT test to test non-stationary input sequences, still, inaccurate results are obtained for the non-stationary input sequence case. Thus, the NIST tests [18], do not supply satisfying results for non-stationary input sequences.

The various tests from NIST [18] supply us a probabilistic confirmation that the tested sequence is random. But it does not give us the location of the periodic part that turns the whole sequence to be non-random. Thus, in cases where the tested sequence failed to be random, the whole sequence is disqualified even if only a short part of it, caused it.

In this paper, we propose a time-frequency approach, that analysis the input sequence in the frequency and time domain in parallel, by using innovative functions such as the Wigner Distribution [26] [27] and the GGD [28] [29]. This new approach solves two open issues:

1) For the first time, the ability to test non-stationary sequences is possible.

2) For the first time, it is possible to locate the periodic section in the tested input sequence.

This approach offers a test that can replace the DFT test in the NIST document [18] and also may be a replacement for test number seven and eight from the NIST tests [18]. The proposed algorithm (test) also finds the location of the periodic part in the tested sequence if it is present. Thus, the periodic part can be cut out from the tested sequence or it can be maybe corrected. Obviously, the corrected and shortened sequences have to be tested again for randomness. The whole issue of how correcting the sequence will not be discussed in this paper. Simulation results will confirm the effectiveness of the proposed approach for non-stationary sequences compared to tests six, seven and eight from the NIST tests [18].

This paper is organized as follows. In Section 2, we explain the motivation of using the Wigner function in estimating the probability of signal randomness degree. In Section 3, we present the various parameters used for the Wigner test which will be shown in Section 4 to be part of the whole proposed algorithm. In Section 4, we present the structure of the whole algorithm, by using both the Wigner and the GGD functions. In Section 5, we test our new proposed algorithm via simulation and compare the results to those obtained from the NIST (test six, seven and eight [18]). Finally, the conclusion is given in Section 6.

2. Motivation

The Wigner function [27] is the Fourier transform for the Autocorrelation function which allows us to get the spectrum as a function of time. In this way we can track the locations of cyclic parts in the tested sequence, by detecting frequency deviations at certain time points. The result of the Wigner function [27] is a matrix, where the lines represent the time domain, the columns represent the frequency domain.

The absolute output values from the Wigner function [27] can be shown in a 3-D image which shows the tested signal passed via the Wigner function [27] as a function of frequency and time. In **Figure 1**, the absolute output values from the Wigner function [27] are shown in a 3-D image, where the input signal to the Wigner function [27] is a cosine signal with a frequency of 10 [MHz]. Please note, that in the time domain the signal at the specified frequency of 10 [MHz] is present throughout the time being tested, while in the frequency domain it has a single frequency. **Figure 2** shows the absolute output values from the Wigner function [27] in a 3-D image, where the input signal to the Wigner function [27] is a combination of three cosine signals with frequencies of 3, 7 and 10 [MHz]. Please note that according to **Figure 2** we have more than only three frequencies due to the cross-correlation effect between the input frequencies. Thus, for signals containing multiple frequencies, the Wigner function [27] creates a difficulty in identifying the frequencies of the input signal. Thus, making a difficulty in identifying the right image (frequency).

The random signal has no specific frequency, otherwise, the signal is not random. In fact, the random signal behaves like the assembly of multiple signals, which causes some difficulties in identifying processes within the signal. **Figure 3** shows a 3-D image obtained from the absolute output values from the Wigner function [27] for a random input sequence.

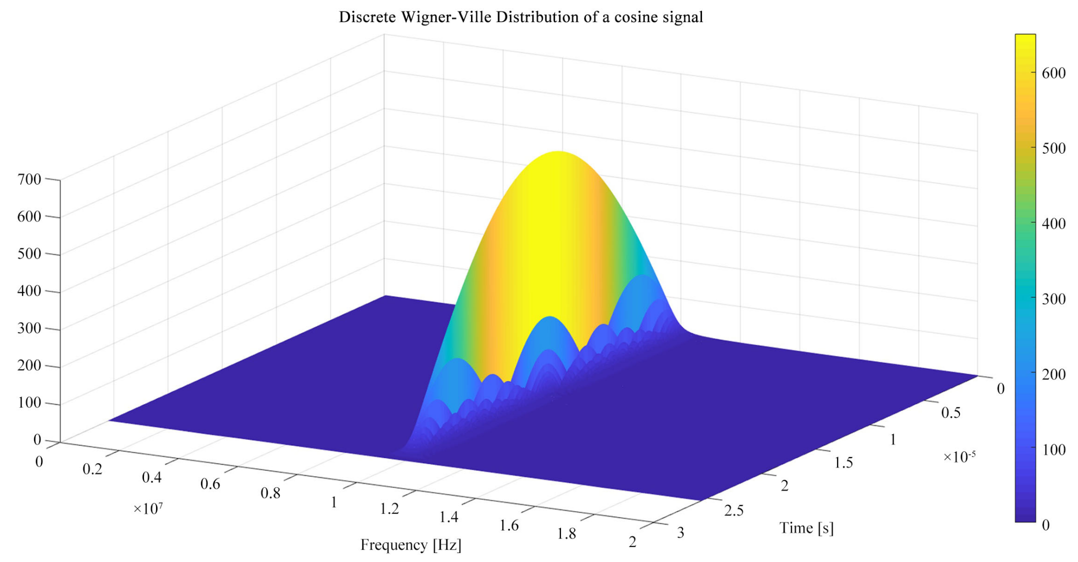


Figure 1. Mesh—one cosine signal, signal frequency: $f_n = 10[\text{MHz}]$, Sampling Frequency: $f_s = 4f_n$, $T = \frac{1}{f_s} = 2.5\times10^{-8}[\sec]$. Length of the signal: 1024 samples $(1024\times T = 2.56\times10^{-5}[\sec])$.

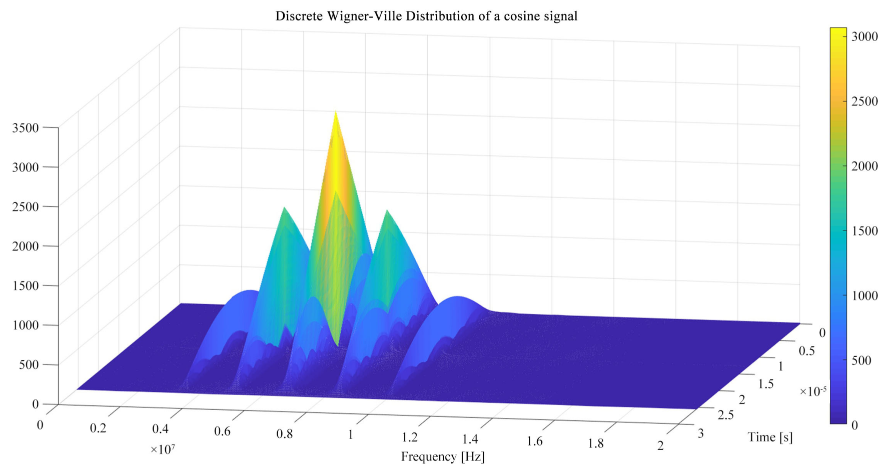


Figure 2. Mesh—two cosine signal, signal frequency: $f_{n_{1,2,3}} = 3,7,10[\text{MHz}]$, sampling frequency: $f_s = 40[\text{MHz}]$, length of the signal: 1024 samples.

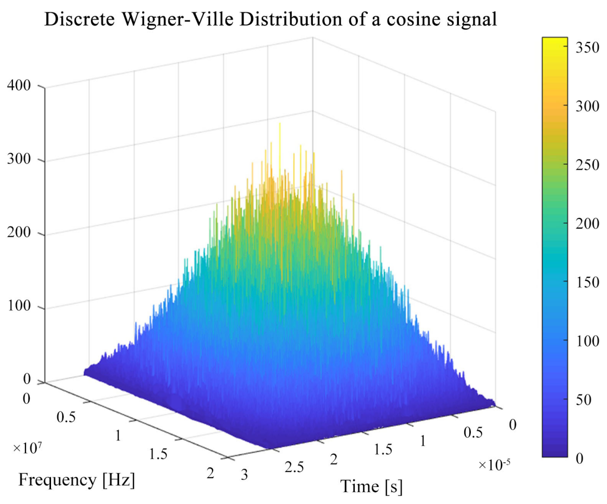


Figure 3. Mesh—random signal, length of the signal: 1024 samples.

Our new method should respond to changes in the time domain, where the existing methods (tests) (NIST test [18]) fail. Thus, we test our method with non-stationary random signal. Non-stationary signals can be artificially assembled by using a number of random signals, as shown in **Figure 4**. where the "cyclic" part is also a random signal that is repeated during the signal.

Figure 5 shows two images obtained from the absolute output values from the Wigner function [27] where the left image was obtained for a random input signal while the right image was obtained for a non-random input signal having periodic parts in the signal. According to **Figure 5**, the signal intensity (z-axis) for the non-random input case, is approximately double the size compared with the signal intensity for the random input case. This shows that the values in the matrix obtained by the Wigner function [27] may be helpful in conjunction with some predefined parameters for classifying the probability of signal randomness degree.

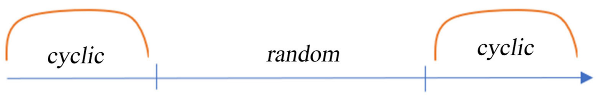


Figure 4. Non-stationary signal structure.

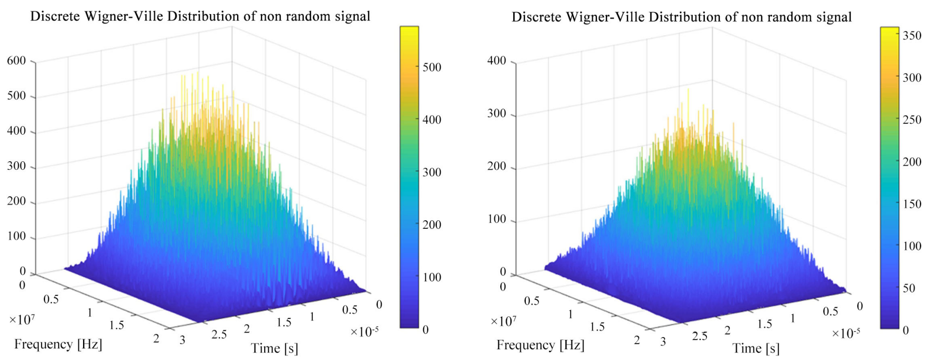


Figure 5. Comparison of 2 tested sequences. Left: non-random sequence, right: random sequence.

With the help of the Wigner function [27] we can detect changes in the time plane. In order to see this, please refer to **Figure 6**, in which we show three different cases where the periodic part is placed in the tested sequence at different places: at the beginning, at the center and at the end of the tested sequence. Please note that in **Figure 6** the y-axis shows the intensity of the signal. According to **Figure 6**, the location of the periodic segment can be detected via the matrix obtained from the output of the Wigner function [27], by tracing the changes in the intensity (y-axis in **Figure 6**).

3. The Wigner Test

In the previous section, we have mentioned that the output values from the matrix obtained from the Wigner function [27] may be helpful in conjunction with some predefined parameters for classifying the probability of signal randomness degree. In the following, we normalized the absolute values from the matrix obtained by the Wigner function [27] by a predefined parameter that depends on the input sequence length. For a sequence length of 1024, 2048 and 4096 the predefined parameter is set to 350, 550 and 850 respectively. These parameters were determined according to signal intensity (z-axis) expected to be obtained from a random sequence, so that a division of this value normalizes the matrix in such a way that it is possible to detect increases in the signal intensity, when the sequence is not random. The value was obtained by performing simulation experiments on 20 random sequences.

In the following we present four parameters ("Power", "Distance", "Density" and "Highest") with their range of values for which the input sequence is declared as a random sequence. Please note, in order to declare that the sequence is a random sequence, the four parameters ("Power", "Distance", "Density" and "Highest") must be in the range for which the sequence is declared as a random

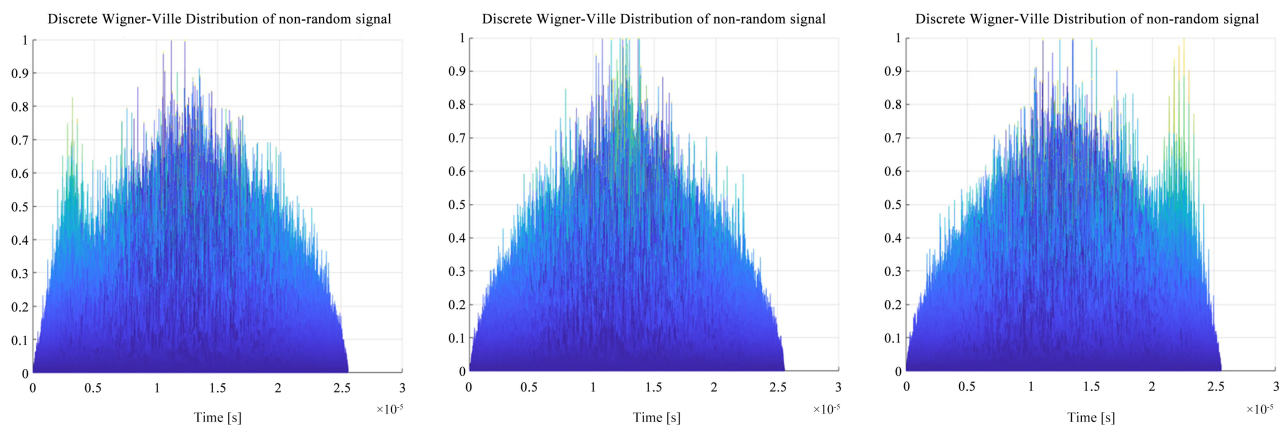


Figure 6. Three tested sequences with a periodic section—a look at the time plane, the location of a periodic segment at the beginning of the sequence (left side), middle (center) and end (right side).

sequence. Please note that in the following we mean by "samples" the absolute values from the matrix obtained from the Wigner function [27]. It should be pointed out that all the values or range of values of the listed parameters from below, were found by simulation experiments only.

The parameters are:

1) Power—Describes the height of the samples. To obtain a normalized range—the samples are divided by their maximum obtained value. For the random signal case, the received range must be between: Power = 0.7 - 0.8.

2) Distance—Describes the "distance" and is equal to the number of samples between two samples with "Power" 0.8. To obtain a normalized range—the received value is divided by the number of samples of the tested sequence. In the random signal case, the received "Distance" is: Distance = 0.1 - 0.3.

3) Density—Describes the ratio of the number of samples above the "Power" of 0.75 in the central part of the matrix to the numbers of samples above the "Power" of 0.75 outside the central part of the matrix. For a random signal, a high ratio is being received, meaning that most of the high samples are in the centre of the signal. In order to declare the input sequence as a random sequence, the received "Density" should be: Density > 0.87.

4) Highest—Describes the number of samples above the "Power" of 0.9. In a random signal the received number is Highest < 15.

Simulation Wigner Test

Table 1 shows some simulation results using the Wigner test (with the above four parameters test). For this purpose, eight sequences were applied with different number of cycles, with different cycle length and segment position of the cyclic part. Each structure was tested with sequence length of 1024, 2048 and 4096. In the following, the highlighted red section is the periodic part.

The sequences structure:

For sequence length of 1024 samples:

Sequence number 1—8 × 128 (8 cycles of 128 bits sequence),

Table 1. Result of the Wigner function.

	Length of sequence	Wigner test				
		Power	Distance	Density	Highest	Random
1	1024	0.9 - 1	0.78	0.51	1208	no
2	1024	0.9 - 1	0.72	0.56	3611	no
3	1024	0.8 - 0.9	0.3	0.78	18	no
4	1024	0.7 - 0.8	0.25	0.88	6	yes
5	1024	0.8 - 0.9	0.39	0.79	19	no
6	1024	0.7 - 0.8	0.3	0.9	19	no
7	1024	0.9 - 1	0.47	0.83	36	no
8	1024	0.9 - 1	0.44	0.76	33	no
9	2048	0.9 - 1	0.75	0.6	3282	no
10	2048	0.9 - 1	0.8	0.5	9182	no
11	2048	0.7 - 0.8	0.22	0.77	13	no
12	2048	0.7 - 0.8	0.26	0.61	7	no
13	2048	0.9 - 1	0.29	0.91	48	no
14	2048	0.7 - 0.8	0.3	0.91	13	yes
15	2048	0.9 - 1	0.4	0.84	43	no
16	2048	0.9 - 1	0.4	0.8	51	no
17	4096	0.9 - 1	0.83	0.62	6346	no
18	4096	0.9 - 1	0.7	0.51	24,527	no
19	4096	0.7 - 0.8	0.25	0.72	4	no
20	4096	0.7 - 0.8	0.3	0.6	11	no
21	4096	0.9 - 1	0.42	0.92	84	no
22	4096	0.7 - 0.8	0.35	0.93	19	no
23	4096	0.9 - 1	0.48	0.84	96	no
24	4096	0.9 - 1	0.52	0.81	124	no

Sequence number 2—16 × 64 (16 cycles of 64 bits sequence),
Sequence number 3—768 + 32 × 8 (periodic part in the end of the sequence),
Sequence number 4—32 × 8 + 768 (periodic part in the start of the sequence),
Sequence number 5—384 + 32 × 8 + 384 (periodic part in the middle of sequence),
Sequence number 6—128 + 320 + 128 + 320 + 128,
Sequence number 7—64 + 128 + 64 + 128 + 64 + 128 + 64 + 128 + 64 + 128 + 64,
Sequence number 8—32 × 2 + 128 + 32 × 2 + 128 + 32 × 2 + 128 + 32 × 2 + 128 + 32 × 2 + 128 + 32 × 2.
For sequence length of 2048 samples:

Sequence number 9—8 × 256 (8 cycles of 256 bits sequence),

Sequence number 10—16 × 128 (16 cycles of 128 bits sequence),

Sequence number 11—1536 + 64 × 8 (periodic part in the end of the sequence),

Sequence number 12—64 × 8 + 1536 (periodic part in the start of the sequence),

Sequence number 13—768 + 64 × 8 + 768 (periodic part in the middle of sequence),

Sequence number 14—256 + 640 + 256 + 640 + 256,

Sequence number 15—128 + 256 + 128 + 256 + 128 + 256 + 128 + 256 + 128 + 256 + 128,

Sequence number 16—64 × 2 + 256 + 64 × 2 + 256 + 64 × 2 + 256 + 64 × 2 + 256 + 64 × 2 + 256 + 64 × 2.

For sequence length of 4096 samples:

Sequence number 17—8 × 512 (8 cycles of 512 bits sequence),

Sequence number 18—16 × 256 (16 cycles of 256 bits sequence),

Sequence number 19—3072 + 128 × 8 (periodic part in the end of the sequence),

Sequence number 20—128 × 8 + 3072 (periodic part in the start of the sequence),

Sequence number 21—1536 + 128 × 8 + 1536 (periodic part in the middle of sequence),

Sequence number 22—512 + 1280 + 512 + 1280 + 512,

Sequence number 23—256 + 512 + 256 + 512 + 256 + 512 + 256 + 512 + 256 + 512 + 256,

Sequence number 24—128 × 2 + 512 + 128 × 2 + 512 + 128 × 2 + 512 + 128 × 2 + 512 + 128 × 2 + 512 + 128 × 2

Simulation results:

Please note, in Table 1 the red colored values are the indices that classify the sequence as non-random. The sequence is considered as random only when the four parameters ("Power", "Distance", "Density" and "Highest") meet the criteria of the random sequence.

According to Table 1, the four parameters ("Power", "Distance", "Density" and "Highest") are effective in classifying if the given sequence is a random sequence or not. But it should be pointed out here that the good results were obtained only for those sequences that have a high periodicity within the sequence or have a long cyclic segment. In addition, with the four parameters ("Power", "Distance", "Density" and "Highest") we are not able to locate the cyclic section in the tested sequence. Thus, as will be explained in the next section we need the GGD [28] [29].

4. The Proposed Algorithm

The general structure of the algorithm can be seen in the block diagram presented in **Figure 7**. In general, the tested sequence is a vector containing only ones and minus ones in the digital time domain. By using the Wigner function [27] this vector is converted to a quadratic matrix which represents the sequence behavior in the time and frequency domains. This matrix allows us to decide whether the tested sequence has a high periodicity within the sequence and should be announced as a non-random sequence, or at this stage, no periodicity was detected within the sequence, thus we should move on to the next step of the algorithm. To detect changes over the time domain, the GGD [28] [29] is being used as will be explained later on in this section.

According to **Figure 7**, $s[n]$ is the sampled output from the RNG with sample rate of f_s, where $T = 1/f_s$ and T is the sampling period $(s(nT) \to s[n])$. Next, this signal $(s[n])$ is converted to an analytic signal with the help of the Hilbert transform [30]. Thus now, the analytical signal $s'[n]$ is driven to the Wigner function [27]. Please note, the input to the Wigner function [27] must be an analytical signal to avoid the aliasing problem with the Wigner transform [27] for non-analytical signals. The output from the Wigner function [27] is $W[n,\omega]$ and is given by:

$$W[n,\tau] = fft\left(R_{ss}[\tau,n]\right) \tag{1}$$

where $\omega = 2\pi f$, f stands for the frequency, $fft(.)$ is the Fast Fourier Transform on $(.)$ and $R_{ss}[\tau,n]$ is the Autocorrelation of $s'[n]$ given by:

$$R_{ss}[\tau,n] = E\left[\left(s'[n] * s'^{*}[n+\tau]\right)\right] \tag{2}$$

where "*" is the multiplication operation, $s'^{*}[n]$ is the conjugate part of $s'[n]$ and $E[.]$ stands for the expectation operation.

At this point, the decision part of **Figure 7** which receives the output from the Wigner function [27] is used to decide whether the incoming sequence has a high periodicity within the sequence or whether the repetitive sequence has a high length. The purpose of this examination is to determine whether the sequence can be disqualified for being a random sequence or can be moved on to the next step of the algorithm. In other words, if the algorithm does not detect a high periodicity within the sequence or a high length of a repetitive sequence at this point, we continue to the next step in the algorithm which is the GGD function [28] [29]. This step is based on examining the changes in the shape parameter associated with the GGD function [28] [29].

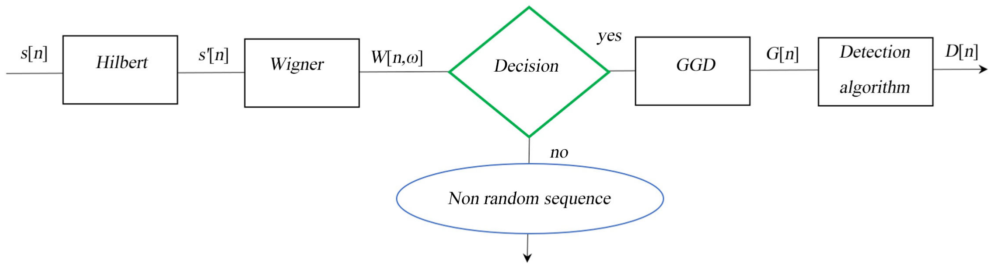


Figure 7. Block diagram of the system.

Changes in the shape parameter of the GGD [28] [29] presentation, change the shape of the probability density function (pdf) of the given input sequence to the GDD [28] [29] which may have a Laplacian, or double exponential distribution, a Gaussian distribution or a uniform distribution for a shape parameter equal to one, two and infinity respectively. Obviously, changes in the pdf of the tested input sequence to the GDD [28] [29], indicates that the input sequence is not random. The shape parameter allows us to characterize the absolute values from the matrix obtained from the Wigner function [27], so that each row or column in the matrix can be expressed by a single value. Thus, a vector that characterizes the time or the frequency domain through the shape parameter can be obtained. In our algorithm, the time domain is in our interest, therefore we calculate the shape parameter on each column (representing the frequency domain) in the matrix. The obtained vector will be denoted as $G[n]$ (**Figure 8**).

According to the GGD function [28] [29], the shape parameter is given by:

$$p(k_i)=\begin{cases} 2^{\ln\frac{27}{16}\Big/\ln\frac{3}{4k_i^2}} & \text{if } k_i\in[0,0.131246) \\ \frac{1}{2a_1}\left(-a_2+\sqrt{a_2^2-4a_1a_3+4a_1k_i}\right) & \text{if } k_i\in[0.131246,0.448994) \\ \frac{1}{2b_3k_i}\left(b_1-b_2k_i-\sqrt{_-i(b_1-b_2k_i)^2-4b_3k_i^2}\right) & \text{if } k_i\in[0.448994,0.671256) \\ \frac{1}{2c_3}(c_2-\sqrt{c_2^2+4c_3\ln\left(\frac{3-4k_i}{4c_1}\right)} & \text{if } k_i\in[0.671256,0.75) \end{cases} \tag{3}$$

where $a_1=-0.535707356,\ a_2=1.168939911,\ a_3=-0.1516189217,$
$b_1=0.9694429,\ b_2=0.8727534,\ b_3=0.07350824,$
$c_1=0.3655157,\ c_2=0.6723532,\ c_3=0.033834.$

According to [28] [29], k_i can be calculated by:

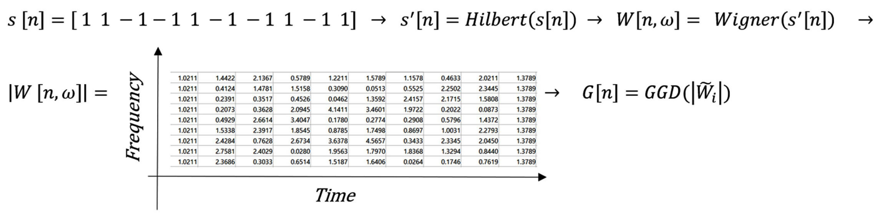
$$k_i=\frac{\left(E\left|\tilde{W}_i\right|\right)^2}{E\left[\left(\left|\tilde{W}_i\right|\right)^2\right]} \tag{4}$$

where $\tilde{W}_i$ is the i-th column (representing the frequency domain) in the matrix $W[n,\omega]$.

Based on Equation (3) and Equation (4) the vector $G[n]$ can be defined:

$$G[n]=\left[p(k_1),p(k_2),p(k_3),\cdots,p(k_N)\right]$$

where N is the length of the tested input sequence.


$s[n]=[1\ 1\ -1-1\ 1\ -1\ -1\ 1\ -1\ 1] \rightarrow s'[n]=Hilbert(s[n]) \rightarrow W[n,\omega]=Wigner(s'[n]) \rightarrow$
$|W[n,\omega]|=$
Frequency
Time
$\rightarrow G[n]=GGD(|\tilde{W}_i|)$

1.0211	1.4422	2.1367	0.5789	1.2211	1.5789	1.1578	0.4633	2.0211	1.3789
1.0211	0.4124	1.4781	1.5158	0.3090	0.0513	0.5525	2.2502	2.3445	1.3789
1.0211	0.2391	0.3517	0.4526	0.0462	1.3592	2.4157	2.1715	1.5808	1.3789
1.0211	0.2073	0.3628	2.0945	4.1411	3.4601	1.9722	0.2022	0.0873	1.3789
1.0211	0.4929	2.6614	3.4047	0.1780	0.2774	0.2908	0.5796	1.4372	1.3789
1.0211	1.5338	2.3917	1.8545	0.8785	1.7498	0.8697	1.0031	2.2793	1.3789
1.0211	2.4284	0.7628	2.6734	3.6378	4.5657	0.3433	2.3345	2.0450	1.3789
1.0211	2.7581	2.4029	0.0280	1.9563	1.7970	1.8368	1.3294	0.8440	1.3789
1.0211	2.3686	0.3033	0.6514	1.5187	1.6406	0.0264	0.1746	0.7619	1.3789

Figure 8. Signal flow. $\tilde{W}_i$ is the i-th column (representing the frequency domain) in the matrix.

The vector $G[n]$ holds all the changes in the time domain of the shape parameter. An example of the shape parameter vector $G[n]$ can be seen in **Figure 9** for a random input sequence of length 1024.

To simplify the shape parameter signal classification, an average operation is carried out on $G[n]$ by averaging every eight following samples of $G[n]$ for an input sequence with length of 1024. In generally, the number of samples for the average operation is 8, 16 and 32 for an input sequence with length of 1024, 2048 and 4096 respectively. The averaged signal $G[n]$ from **Figure 9** is presented in **Figure 10**. Please note, in order to maintain the same vector length, each of these eight samples (for an input sequence with length of 1024) received the averaged value obtained by the averaging operation.

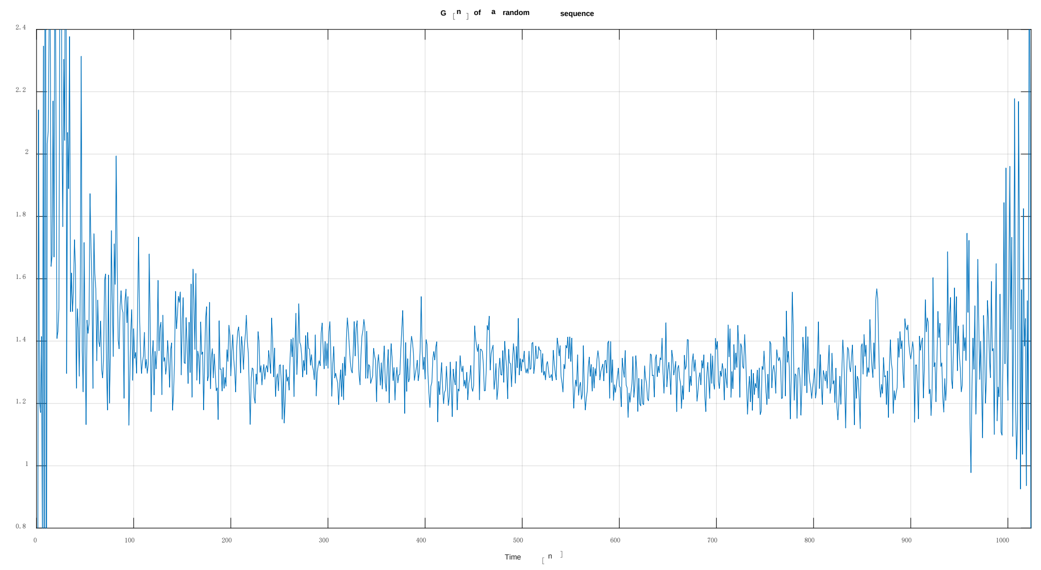


Figure 9. The vector $G[n]$. n represents the sample number.

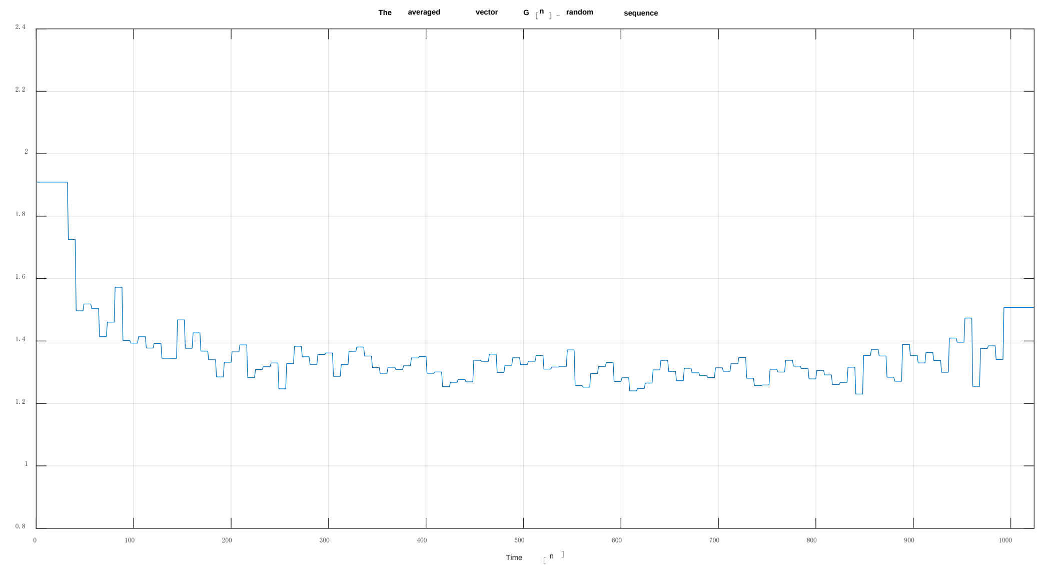


Figure 10. Average of $G[n]$ (an average of 8 samples).

In the following we define:

$$\tilde{W}_i = W(i,j) \text{ for } i = 1,2,3,\cdots,N \ (\text{time domain}) \tag{5}$$

$$\hat{W}_j = W(i,j) \text{ for } j = 1,2,3,\cdots,N \ (\text{freguency domain}) \tag{6}$$

According to Equation (4), Equation (5) and Equation (6) we can define:

$$k_i = \frac{\left(E\left|\tilde{W}_i\right|\right)^2}{E\left[\left(\left|\tilde{W}_i\right|\right)^2\right]} \text{ for } i = 1,2,3,\cdots,N \ (\text{time domain}) \tag{7}$$

$$\hat{k}_j = \frac{\left(E\left|\hat{W}_j\right|\right)^2}{E\left[\left(\left|\hat{W}_j\right|\right)^2\right]} \text{ for } j = 1,2,3,\cdots,N \ (\text{freguency domain}) \tag{8}$$

Therefore, the vectors $G[n]$ and $\hat{G}[n]$ can be defined:

$$G[n] = \left[p(k_1), p(k_2), p(k_3), \cdots, p(k_N)\right] \ (\text{time domain}) \tag{9}$$

$$\hat{G}[n] = \left[p(\hat{k}_1), p(\hat{k}_2), p(\hat{k}_3), \cdots, p(\hat{k}_N)\right] \ (\text{freguency domain}) \tag{10}$$

After the average operation on $G[n]$ and $\hat{G}[n]$, we get the vectors $\tilde{g}[n]$ and $\hat{g}[n]$ respectively. In the following we set q equal to 1, 2, 4 for an input sequence length of 1024, 2048 and 4096 respectively.

$$\tilde{g}[n] = \left[a_0, a_{8q}, a_{16q}, a_{24q}, \cdots, a_{N-8q}\right] \ (\text{time domain})$$

$$\hat{g}[n] = \left[\hat{a}_0, \hat{a}_{8q}, \hat{a}_{16q}, \hat{a}_{24q}, \cdots, \hat{a}_{N-8q}\right] \ (\text{freguency domain})$$

For simplicity we show in the following a_j and $\hat{a}_j$ for $q = 1$ where $j = 0, 8q, \cdots, N - 8q$. Please refer to the **Appendix** for the definition of a_j and $\hat{a}_j$ for q equal to two and four.

$$a_j = \left[\frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \right.$$
$$\left. \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j], \frac{1}{8}\sum_{i=1}^{i=8} G[i+j]\right]$$

$$\hat{a}_j = \left[\frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \right.$$
$$\left. \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j], \frac{1}{8}\sum_{i=1}^{i=8} \hat{G}[i+j]\right]$$

Detection Algorithm

The signal $D[n]$ is the output from the detection algorithm (**Figure 7**), that classifies whether the input sequence is a random sequence or not, based on passing or not passing seven test cases that will be presented in this section. In addition, this algorithm marks the places that are suspicious of having a repeated sequence within the tested sequence. Thus, there are two steps in the detection algorithm:

1) Determine whether the sequence is random or not random.

2) Finding the periodic part in the tested input sequence if it exists.

Let us start explaining the first step. Here we apply the histogram function (from Matlab version R2018a) on the vectors $\tilde{g}[n]$ and $\hat{g}[n]$ (obtained from the tested input sequence) in order to have the distribution of the appearance of the values of the shape parameter. In addition, for comparison, we apply also the histogram function on the vectors $\tilde{g}[n]$ and $\hat{g}[n]$ obtained from a random sequence. In order to generate this random sequence, the "randn" function from the Matlab version R2018a is applied which generates "random" numbers from a Gaussian distribution with variance of one and with a zero mean. Next, if the generated number is positive or equal to zero, then the used number is set to one, else it is set to zero. Please note that for some tests in the NIST test [18] a sequence of ones and zeros are required. Thus, our generated sequence is converted to ones and zeros following [18]. It should be pointed out that the various parameters for the following seven test cases were found by simulation experiments only. In the following, we use thirty bins for representing the histogram. But those bins that have a height less than 0.01 will be deleted and not considered in the following tests. In addition, after the deletion of those bins that have a height of less than 0.01, the bins that are not close to the center bins are also deleted. In order to declare a sequence to be a random sequence, the seven test cases have to be passed.

Please note that in **Figures 11-17** the x-axis represents the various values of the shape parameter associated with the tested input sequence, while the y-axis represents the probability density of their appearance. As already was mentioned earlier in this paper, the shape parameter allows us to characterize the absolute values from the matrix obtained from the Wigner function [27], so that each row or column in the matrix can be expressed by a single value. Thus, a vector that characterizes the time or the frequency domain through the shape parameter can be obtained.

In the following, we mean by saying that the test is performed in the time domain that the shape parameter was calculated on each column (representing the frequency domain) in the matrix. In addition, if it is stated that the test is performed in the frequency domain, it means that the shape parameter was calculated on each row (representing the time domain) in the matrix.

The seven test cases are:

Test 1—Checks, whether the maximum height is at the center of the graph and if the width of the graph is smaller than 0.2. This test is performed in the time domain. If the conditions are not met—the signal will be marked as a non-random sequence. An example of this test can be seen in **Figure 11**, where at the left figure we have the result for a random sequence while at the right figure we have the result for a non-random input sequence. According to **Figure 11**, the width of the graph in the right figure is higher than 0.2 and the maximum height is not at the center of the graph. Thus, the tested input sequence is suspected to be a non-random sequence. Please note, that we have marked all the bins that were already deleted in the early stage of our test.

Test 2—Checks if there is more than one peak in the graph. It looks for the two highest bins in the graph and declares that the tested input sequence is suspected to be a non-random sequence if at least there is one bin between the two highest found bins, and the difference between the heights of those two highest bins is less than 15%. This test is performed in the time domain. An example for this test can be seen in Figure 12 where at the right figure we have the result for a non-random input sequence.

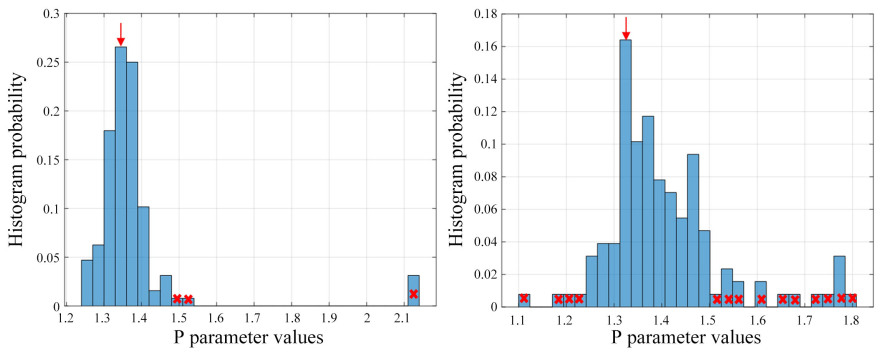


Figure 11. Example for test 1. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

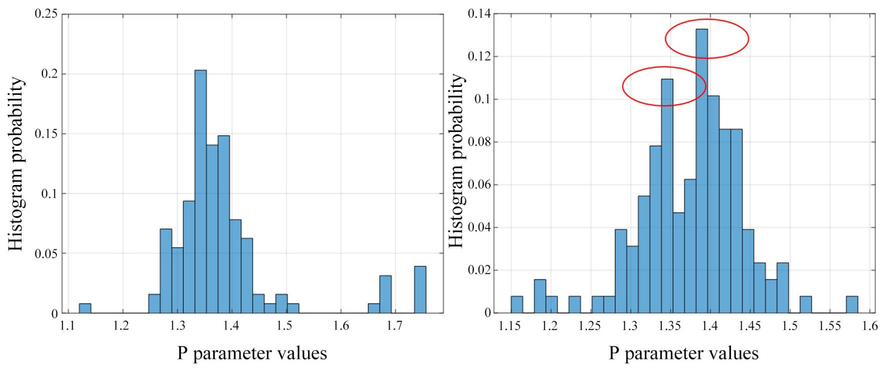


Figure 12. Example for test 2. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

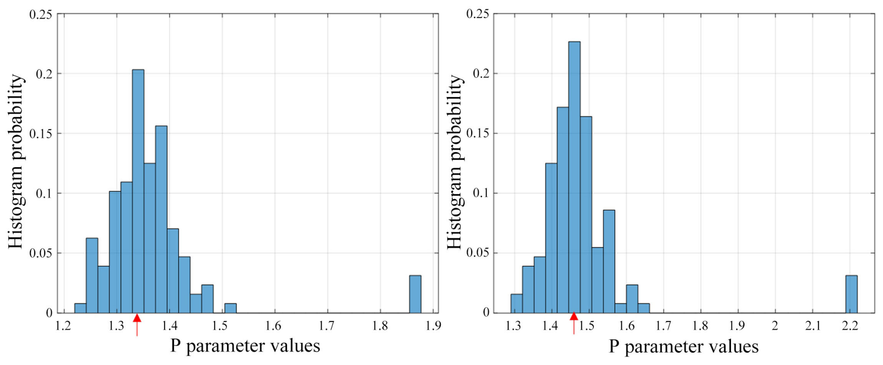


Figure 13. Example for test 3. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

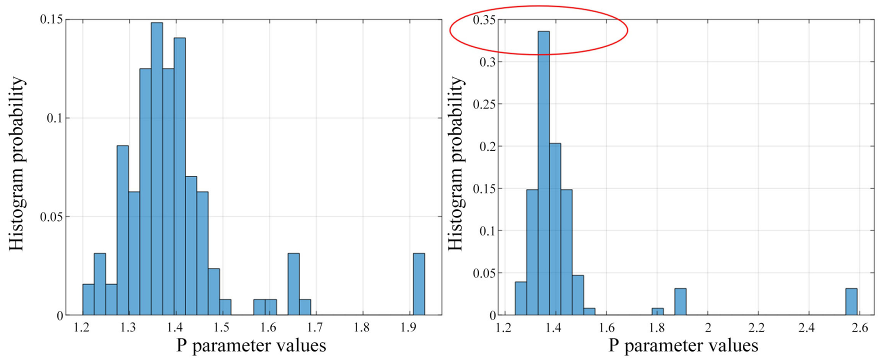


Figure 14. Example for test 4. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

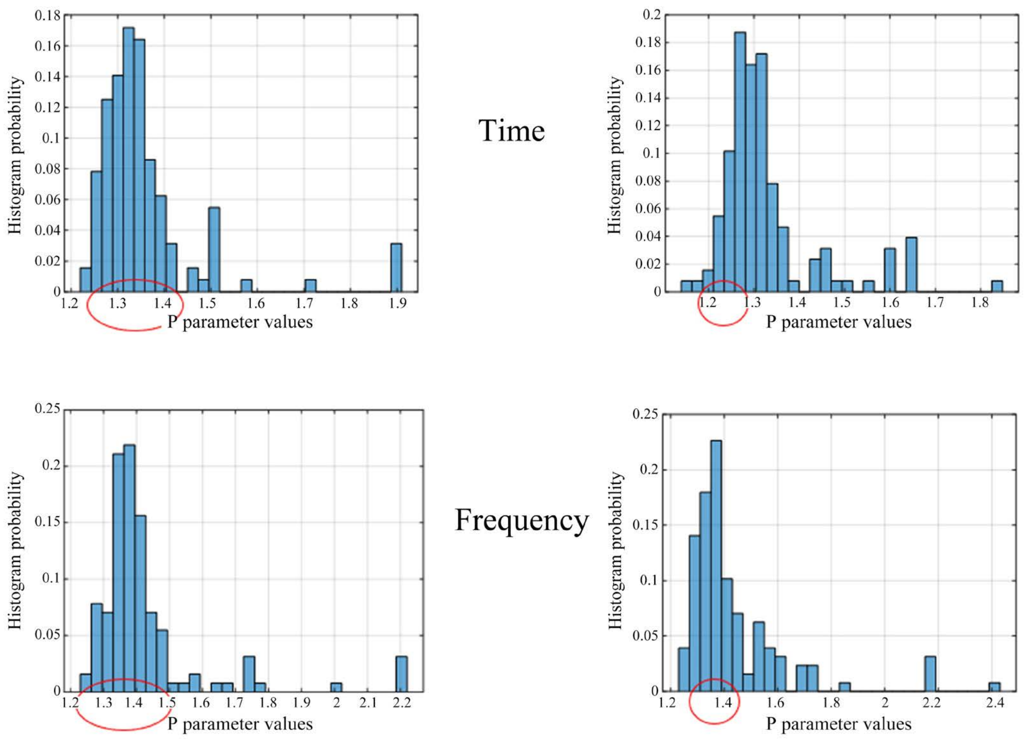


Figure 15. Example for test 5. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

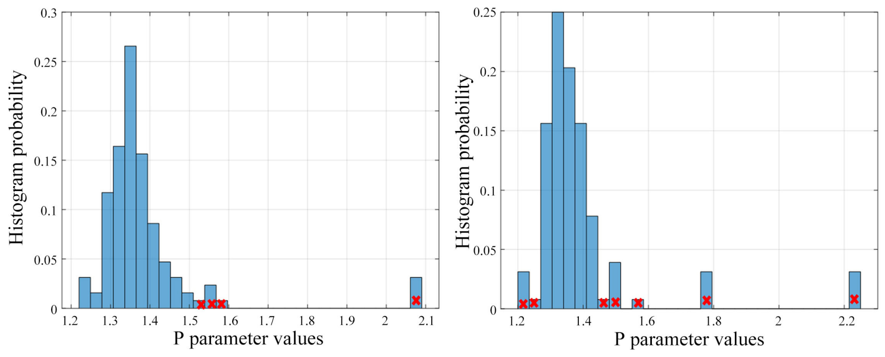


Figure 16. Example to test 6, the histogram in the left is random sequence and on the right is non-random sequence.

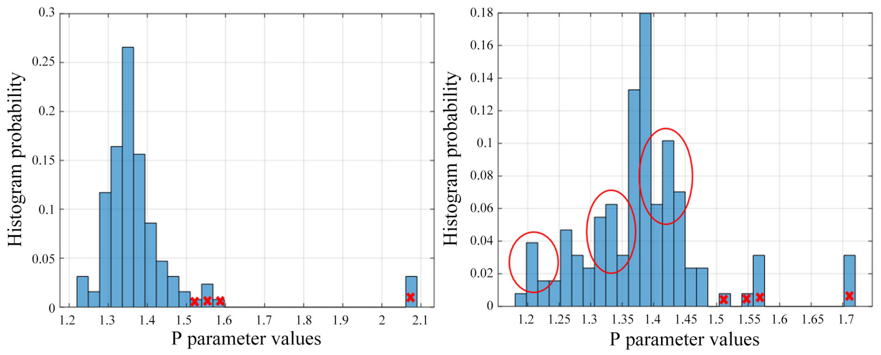


Figure 17. Example to test 7. The histogram in the left is for a random sequence and on the right is for a non-random sequence.

Test 3—Checks whether the maximum height is obtained in the x-axis between 1.29 and 1.4. If it is outside of this boundary, the sequence is marked as a non-random sequence. This test performed in the time domain. An example of this test can be seen in Figure 13 where on the right figure we have the result for a non-random sequence where the maximum height is not obtained in the x-axis between 1.29 and 1.4.

Test 4—Checks whether the maximum height is above 0.25. If it is, the tested input sequence is marked as a non-random sequence. This test is performed in the frequency domain. The main idea behind this test is that if there is a concentration around a certain frequency, it indicates that the tested sequence is suspected to have a periodicity within the sequence. An example of this test can be seen in Figure 14, where we have at the right figure the result for a non-random input sequence where the maximum bin is more than 0.25.

Test 5—Checks the position of the maximum height of the bin between the time domain and the frequency domain. If there is no match, this indicates that the tested input sequence is a non-random sequence. An example for this test can be seen in Figure 15, where we have at the right figures the result for a non-random input sequence where there is no match between the position of the maximum height of the bin between the time domain and the frequency domain.

Test 6—This test checks two conditions concerning the heights of the bins. First, it checks if a bin can be found with a height lower than 0.05. Next, it checks whether a bin can be found with a height above 0.2. If no bin is left with a height lower than 0.05 and a bin is found with a height above 0.2, the sequence is announced as a non-random sequence. This test is performed in the time domain. An example for this test can be seen in Figure 16, where we have at the right figure the result for a non-random input sequence where we have marked all the bins that were already deleted before entering into this test. Now, according to Figure 16, we have a bin with a height above 0.2 and there is no bin with a height lower than 0.05. Thus, the result presented in the right figure of Figure 16, corresponds to a non-random sequence. Please note, that we have marked in Figure 16 all the bins that were already deleted in the early stage of our test.

Test 7—The pdf should increase monotonically towards the center bin (or to the bin with the maximum height) of the pdf and decrease monotonically from the center bin (or from the bin with the maximum height) downwards. If there is a 40% decrease in the height of a bin with the bin before it, the tested sequence is announced as a non-random sequence. An example for this test can be seen in Figure 17, where we have at the right figure the result for a non-random input sequence where we have circled the places where the decrease in the height of 40% was found. Please note, that we have marked all the bins that were already deleted in the early stage of our test.

Next, we determine the mathematical description of the above seven tests.

First, we define:

$$\sum_{i=1}^{i=L} \tilde{m}_i = 1; \ \sum_{i=1}^{i=L} \hat{m}_i = 1$$

where $\tilde{m}_i$ and $\hat{m}_i$ holds the number of observations of $\tilde{g}[n]$ and $\hat{g}[n]$ respectively that fall into each of the disjoint bins divided by the length of $\tilde{g}[n]$ and $\hat{g}[n]$ respectively, and L is the total number of bins.

$$x_d(i) = \min(\tilde{g}[n]) + diffx_d * (i-1), \ i = 1,2,3,\cdots,L$$

$$diffx_d = \frac{\max(\tilde{g}[n]) - \min(\tilde{g}[n])}{L}$$

where $x_d(i)$ holds the value in the x-axis of the edge where the bin begins.

$$\hat{x}_d(i) = \min(\hat{g}[n]) + \frac{\max(\hat{g}[n]) - \min(\hat{g}[n])}{L} * (i-1), \ i = 1,2,3,\cdots,L$$

where $\hat{x}_d(i)$ holds the value in the x-axis of the edge where the bin begins. It was already mentioned earlier that those bins that have a height less than 0.01 will be deleted and not considered in the seven tests. In addition, after the deletion of those bins that have a height of less than 0.01, the bins that are not close to the center bins are also deleted. Thus, from the mathematical point of view we may write:

```
Step 1: find the index or indexes where ( $\tilde{m}_i < 0.01$ ) is true and set those
        indexes to $i_1, i_2, i_3, \cdots, i_L$ .
Step 2: set $\tilde{m}_i(i_a) = 0$ ; for $a = 1,2,3,\cdots,L$ according to the founded
        indexes from Step 1.
Step 3: find $\max(\tilde{m}_1, \tilde{m}_2, \cdots, \tilde{m}_L)$ and set the index $i$ where the maximum
        was achieved as $i_1$
Step 4: $a = 1$;
        for $i = 1$ to ( $i_1 - 1$ ) checking if there is a bin with $\tilde{m}_i = 0$ in order to
                              set "$a$" as the start of the histogram after this bin.
            if $\tilde{m}_i == 0$ then
                $a = i + 1$
            end
          end
Step 5: $b = 0$;
```

```
        $i = i_1 + 1$;
        while ( $b == 0$ ) checking if there is a bin with $\tilde{m}_i = 0$ in order to
            set "$b$" as the end of the histogram before this bin.
            if ( $i == L$ ) then
                $b = L$
            end
            if ( $\tilde{m}_i == 0$ ) then
                $b = i - 1$
            end
            $i = i + 1$
    end
```

Test number 1:

```
Step 1: find $\max(\tilde{m}_a, \tilde{m}_{a+1}, \cdots, \tilde{m}_b)$ and set the index $i(a < i < b)$ where the
        maximum was achieved as $i_1$.
Step 2: if ( $|(b - i_1) - (i_1 - a)| \le 2$ & $x_d(b) + diffx_d - x_d(a) < 0.2$ ) then
                $Dn(1) = 1$ Random signal
    else
                $Dn(1) = 0$ Not Random signal
    end
```

Test number 2:

```
Step 1: find $\max(\tilde{m}_a, \tilde{m}_{a+1}, \cdots, \tilde{m}_b)$ and set the index $i(a < i < b)$ where the
        maximum was achieved as $i_1$.
Step 2: find $\max(\tilde{m}_a, \tilde{m}_{a+1}, \cdots, \tilde{m}_{i_1-1}, \tilde{m}_{i_1+1}, \cdots, \tilde{m}_b)$ and set the index $i$
        ( $a < i < b$ ) where the maximum was achieved as $i_2$.
Step 3: if $\left(\frac{\tilde{m}_{i_1} - \tilde{m}_{i_2}}{\tilde{m}_1} 100\right) < 15$ & ( $|i_2 - i_1| \ge 2$ ) then
                $Dn(2) = 0$ Not Random signal
            else
                $Dn(2) = 1$ Random signal
            end
```

Test number 3:

```
Step 1: find $\max(\tilde{m}_a, \tilde{m}_{a+1}, \cdots, \tilde{m}_b)$ and set the index $i$
        ( $a < i < b$ ) where the maximum was achieved as $i_1$.
Step 2: if $1.29 < x_d(i_1) < 1.4$ then
            $Dn(3) = 1$ Random signal
    else
            $Dn(3) = 0$ Not Random signal
    end
```

Test number 4:

```
Step 1: if $(\max(\hat{m}_1, \hat{m}_2, \cdots, \hat{m}_L)) > 0.25$ then
            $Dn(4) = 0$ Not Random signal
    else
            $Dn(4) = 1$ Random signal
```

```
        end
Test number 5:
Step 1: find  $\max(\tilde{m}_a,\tilde{m}_{a+1},\cdots,\tilde{m}_b)$  and set the index $i$
            ( $a < i < b$ ) where the maximum was achieved as  $i_1$.
Step 2: find  $\max(\hat{m}_1,\hat{m}_2,\cdots,\hat{m}_L)$  and set the index $i$
            ( $1 < i < L$ ) where the maximum was achieved as  $i_2$.
Step 3: if  $|x_d(i_1)-\hat{x}_d(i_2)| \le 0.05$  then
              $Dn(5)=1$  Random signal
        else
              $Dn(5)=0$  Not Random signal
        end
Test number 6:
Step 1: if ( $\max(\tilde{m}_a,\tilde{m}_{a+1},\cdots,\tilde{m}_b) > 0.2$ ) & ( $\min(\tilde{m}_a,\tilde{m}_{a+1},\cdots,\tilde{m}_b) > 0.05$ )
        then
              $Dn(6)=0$  Not Random signal
        else
              $Dn(6)=1$  Random signal
        end
Test number 7:
Step 1: find  $\max(\tilde{m}_a,\tilde{m}_{a+1},\cdots,\tilde{m}_b)$  and set the index $i$
              ( $a < i < b$ ) where the maximum was achieved as  $i_1$.
Step 2:  $Dn(7)=1$  Random signal
         for  $i=a$  to  $i_1$  checking if there is an increase monotonically
              towards the center bin
            if ( $\frac{|\tilde{m}_i-\tilde{m}_{i+1}|}{\tilde{m}_i}100 \ge 40$ ) & ( $\tilde{m}_i > \tilde{m}_{i+1}$ ) then
                 $Dn(7)=0$  Not Random signal
        end
    end
          for  $i=i_1$  to $b$ checking if there is a decrease monotonically
               from the center bin down
             if ( $\frac{|\tilde{m}_i-\tilde{m}_{i+1}|}{\tilde{m}_i}100 \ge 40$ ) & ( $\tilde{m}_i < \tilde{m}_{i+1}$ ) then
                 $Dn(7)=0$  Not Random signal
        end
    end
Outcome from the seven tests:
if  $\sum_{k=1}^{k=7} Dn(k) == 7$  then
 $D=1$  the tested sequence is Random
else
 $D=0$  the tested sequence is not Random
end
```

Please note that when $D=0$ we continue to the next step in the algorithm of locating the periodic part.

The second step in the detection algorithm is to find the periodic part in the tested input sequence. For this purpose, we apply three different kind of averages on the tested averaged vector $G[n]$ ($\tilde{g}[n]$). This part of the algorithm is performed only on $\tilde{g}[n]$ which represents the time domain, because in this stage we want to locate the periodic part. The three different kind of averages are as follows:

1) The average of the whole vector $\tilde{g}[n]$ (except for some samples at the beginning and at the end) is carried out here. Thus, one average value is obtained. In the following, we denote this kind of averaging as "Average number 1".

2) The vector $\tilde{g}[n]$ (except for some samples at the beginning and at the end) is divided into 3 parts (40% - 20% - 40%) and then the average operation is carried out on each part. Thus, here we have three average values. In the following, we denote this kind of averaging as "Average number 2".

3) The vector $\tilde{g}[n]$ (except for some samples at the beginning and at the end) is divided into 3 parts (25% - 50% - 25%) and then the average operation is carried out on each part. Thus, here we have three average values. In the following, we denote this kind of averaging as "Average number 3".

Figure 18 presents the three different kind of averages carried out on the tested averaged vector $G[n]$ ($\tilde{g}[n]$).

where by "Margin" we mean that at the beginning and at the end of the averaged tested $G[n]$ no averaging operation is carried out. For the case of a 1024, 2048 and 4096 length sequence, the margin on each side is 32 samples, 64 samples and 128 samples respectively.

Now, in order to declare an area to be suspicious of not being a random sequence, we compare the values of the averaged vector $G[n]$ to the values of Average number 1, Average number 2 and to Average number 3. In general, if the values of the averaged vector $G[n]$ are above or below the average number (Average number $1 \pm \varepsilon_1$, Average number $2 \pm \varepsilon_2$, Average number $3 \pm \varepsilon_3$ where ε_1, ε_2 and ε_3 are predefined values) for some tested values that come in sequence, a suspicious place is declared. For more details, please refer to Figures 19-21 which describe in more details the comparison operation between the values of the averaged vector $G[n]$ and the Average number 1 to Average number 3.

Figures 22-25 show the simulated results obtained for a non-random sequence with length of 1024 where the averaged vector $G[n]$ ($\tilde{g}[n]$) was compared to the three different kind of averages (Average number 1, Average number 2, Average number 3) while Figure 26 holds the total result (collected from Figure 22, Figure 23, Figure 25) of the suspicious places where the tested sequence is suspicious not to be a random sequence.

In the following, we describe the flowchart for finding the suspicious places of the periodic parts in the tested input sequence by using the values for Average number 1, Average number 2 and Average number 3.

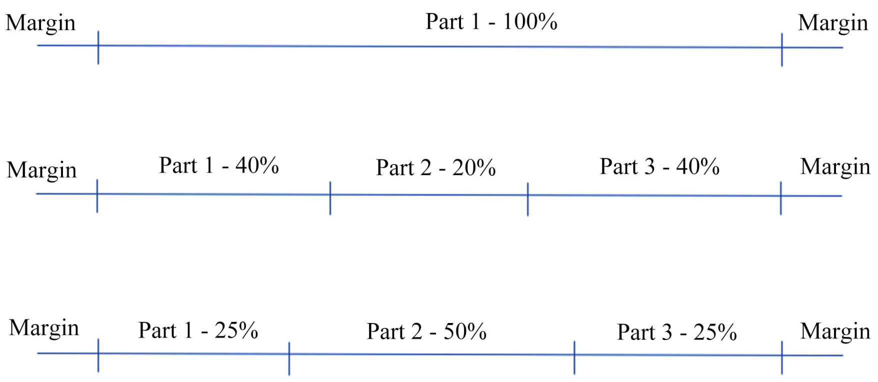


Figure 18. 3 kinds of averages are performed on the sequence.

```
% p_T= the averaged vector G[n].
len_seq=1024; %length of the sequence
edge=32;      %Is a parameter that represents the margin in the input sequence

% p_parameter_time1, p_parameter_time2, p_parameter_time3 are vectors that hold the places in the
%    averaged vector G[n] where the values there are found to be below and above a predefined value
%    associated with "Average_number_1".
%  p_T_output1 is a vector that holds all the suspicious places in the averaged vector G[n] for  not
%    being a random sequence.

p_parameter_time1=find(p_T(edge+1:fix(0.25*len_seq))>average_number_1*1.01);
p_T_output1(p_parameter_time1+edge)=1;

p_parameter_time2=find(p_T(fix(len_seq*0.45):fix(len_seq*0.55))< average_number_1*0.97);
p_T_output1(p_parameter_time2+(fix(len_seq*0.45)))=1;

p_parameter_time3=find(p_T(fix(0.75*len_seq)+1:(len_seq-edge))>average_number_1*1.01);
p_T_output1(p_parameter_time3+(fix(len_seq*0.75)))=1;

% Length_Test_1 = Is a function that checks whether the length of consecutive samples in the input vector
%    is greater than a predefined value. If it is not, the function returns the input vector with those
%    deleted samples.
% edge_short = The value is set to a minimum length relative to the vector p_parameter_time1 or
%    p_parameter_time3. For a sequence length of 1024, we set 'edge_short' = 20.
% center_short = The value is set to a minimum length relative to the vector p_parameter_time2.
%    For a sequence length of 1024, we set 'center_short' = 33

p_T_output1=Length_Test_1(p_T_output1,edge_short,center_short);
```

Figure 19. Matlab code, associated with Average number 1.

```
% p_T= the averaged vector G[n].
len_seq=1024; %length of the sequence

% p_parameter_time4, p_parameter_time5 are vectors that hold the places in the
%    averaged vector G[n] where the values there are found to be above a predefined value
%    associated with "Average_number_2".
%  p_T_output2 is a vector that holds all the suspicious places in the averaged vector G[n] for  not
%    being a random sequence.

p_parameter_time4=find(p_T(fix(0.24*len_seq)+1:fix(0.35*len_seq))>average_number_2*1.002);
p_T_output2(p_parameter_time4+p_T(fix(0.24*len_seq)))=1;

p_parameter_time5=find(p_T(fix(0.65*len_seq)+1:fix(0.76*len_seq))>average_number_2*1.002);
p_T_output2(p_parameter_time5+(fix(len_seq*0.65)))=1;

% Length_Test_2 = Is a function that checks whether the length of consecutive samples in the input vector
%    is greater than a predefined value. If it is not, the function returns the input vector with those
%    deleted samples.
% seq_short1 = The value is set to a minimum length relative to the vector p_parameter_time4 or
%    p_parameter_time5. For a sequence length of 1024, we set 'seq_short1' = 17.

p_T_output2=Length_Test_2(p_T_output2,seq_short1);
```

Figure 20. Matlab code, associated with Average number 2.

```
% p_T= the averaged vector G[n].
len_seq=1024; %length of the sequence

% p_parameter_time6, p_parameter_time7, p_parameter_time8 are vectors that hold the places in the
%    averaged vector G[n] where the values there are found to be above a predefined value
%    associated with "Average_number_3".
%  p_T_output3 is a vector that holds all the suspicious places in the averaged vector G[n] for  not
%    being a random sequence.

p_parameter_time6=find(p_T(fix(0.34*len_seq)+1:fix(0.45*len_seq))>average_number_3);
p_T_output3(p_parameter_time6+p_T(fix(0.34*len_seq)))=1;

p_parameter_time7=find(p_T(fix(0.55*len_seq)+1:fix(0.66*len_seq))>average_number_3);
p_T_output3(p_parameter_time7+(fix(len_seq*0.55)))=1;

p_parameter_time8=find(p_T(fix(0.45*len_seq)+1:fix(0.55*len_seq))>average_number_3*1.05);
p_T_output3(p_parameter_time8+(fix(len_seq*0.45)))=1;

% Length_Test_3 = Is a function that checks whether the length of consecutive samples in the input vector
%    is greater than a predefined value. If it is not, the function returns the input vector with those
%    deleted samples.
% seq_short3 = The value is set to a minimum length relative to the vector p_parameter_time6 or
%    p_parameter_time7. For a sequence length of 1024, we set 'seq_short3' = 17.
% seq_short4 = The value is set to a minimum length relative to the vector p_parameter_time8
%    For a sequence length of 1024, we set 'seq_short3' = 9.

p_T_output3=Length_Test_3(p_T_output3,seq_short3,seq_short4);
```

Figure 21. Matlab code, associated with Average number 3.

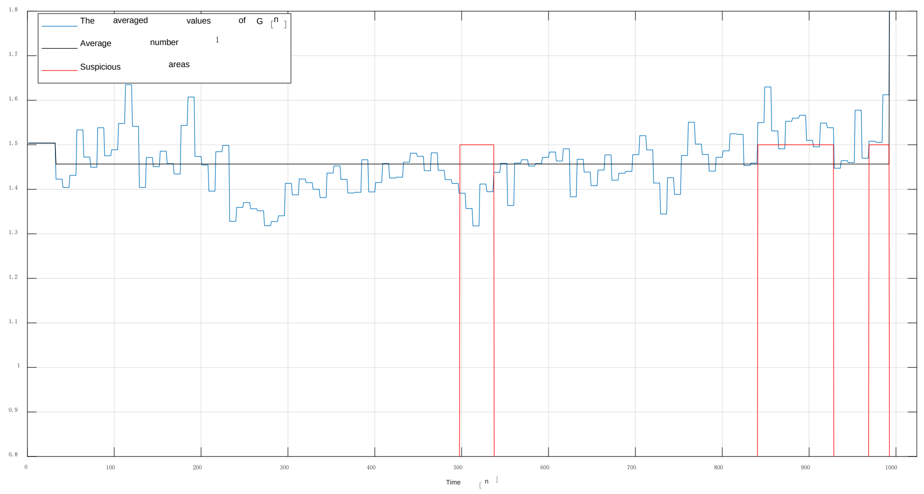


Figure 22. Comparison to Average number 1.

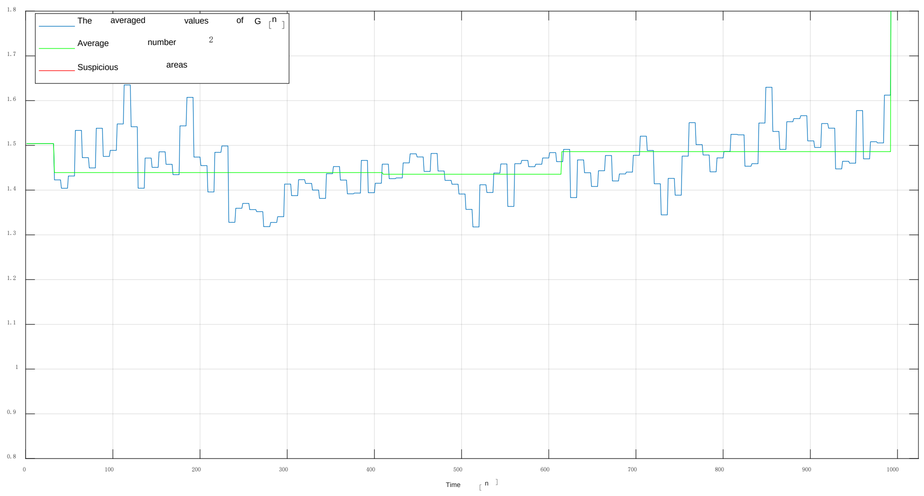


Figure 23. Comparison to Average number 2.

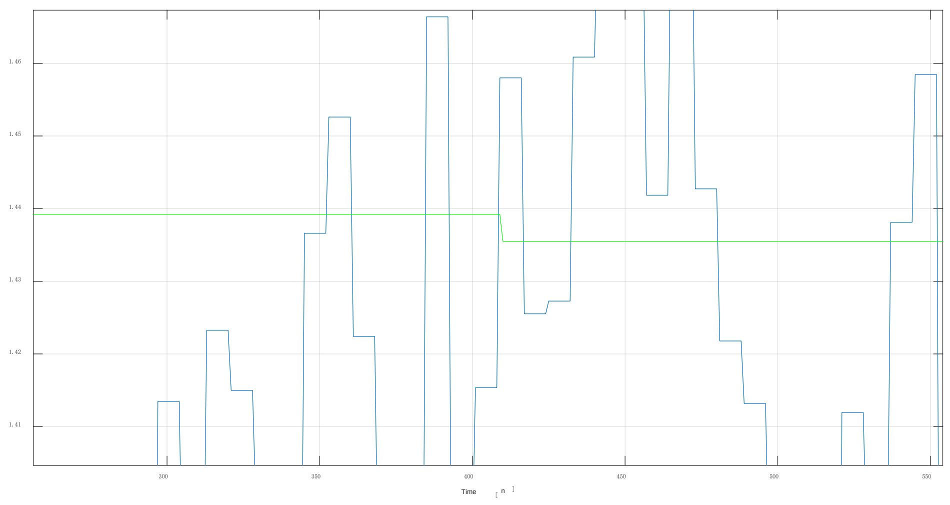


Figure 24. Zoomed in version of Figure 23, please note that we have two different levels at $350 < n < 450$. This cannot be seen in Figure 23 because the two levels are very close.

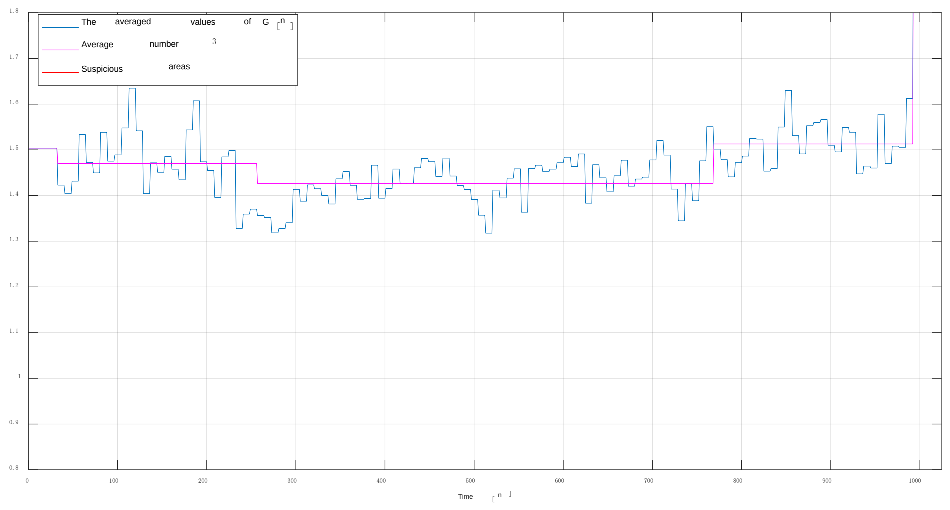


Figure 25. Comparison to Average number 3.

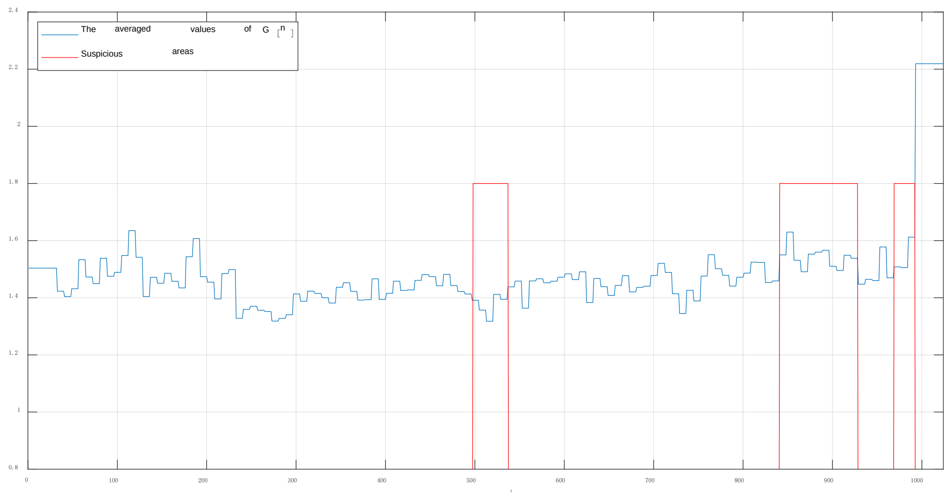


Figure 26. The result after collecting the three images from Figure 22, Figure 23, Figure 25.

<u>**Comparison to Average number 1:**</u>

Step 1: set $g_{average1} = \left[\tilde{g}(edge), \tilde{g}(edge+1), \cdots, \tilde{g}(N-edge)\right]$.

"*edge*" is the length of the margin defined as 32, 64 and 128 for a 1024, 2048 and 4096 length of input sequence respectively.

The input sequence length is defined as *N*.

Step 2: set Average number 1 = $\dfrac{g_{average1}}{N-2*edge}$

Step 3: set $g_{test1_a} = \left[\tilde{g}(edge), \tilde{g}(edge+1), \cdots, \tilde{g}(\lfloor N*0.25 \rfloor)\right]$

where $\lfloor (.) \rfloor$ is the rounding down operation on $(.)$.

Step 4:find the index or indexes where ($g_{test1_a} > 1.01 * Average\ number\ 1$)

is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.25 \rfloor - edge+1}$

Step 5: set $g_{comparison1_a}(i_a) = 1$;

for $a = 1, 2, 3, \cdots, (\lfloor N*0.25 \rfloor - egde+1)$

according to the founded indexes of Step 4.

Step 6: find the number of groups in $g_{comparison1_a}$ of consecutive samples equal to one and set this number to *K*1.

Step 7: find the length of those groups of consecutive samples equal to one from Step 6 and hold them in the vector:

$length1_a(j)$; for $j = 1, \cdots, K1$

Step 8: find the start and end position of the founded groups from Step 6 and hold them in the matrix:

$place1_a(j,i)$; for $j = 1, \cdots, K1$; $i = 1, 2$

Step 9: set $number1_a = K1$

*A*1 = 20, 40 and 80 for 1024, 2048 and 4096 length of sequence respectively.

```
for $j = 1$ to $K1$
        if $length1_a(j) < A1$ then
            $number1_a = number1_a - 1$
            for $i = place1_a(j,1)$ to $place1_a(j,2)$
                $g_{comparison1_a}(i) = 0$
            end
        end
    end
```

Step 10: set $g_{test1_b} = \left[\tilde{g}(\lfloor N*0.45 \rfloor), \tilde{g}(\lfloor N*0.45 \rfloor+1), \cdots, \tilde{g}(\lfloor N*0.55 \rfloor)\right]$

Step 11: find the index or indexes where ($g_{test1_b} < 0.97 * Average\ number1$)

is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.55 \rfloor - \lfloor N*0.45 \rfloor + 1}$

Step 12: set $g_{comparison1_b}(i_a) = 1$;

for $a = 1, 2, 3, \cdots, (\lfloor N*0.55 \rfloor - \lfloor N*0.45 \rfloor + 1)$ according to the founded indexes of Step 11.

Step 13: find the number of groups in $g_{comparison1_b}$ of consecutive samples equal to one and set this number to *K*2.

Step 14: find the length of those groups of consecutive samples equal to one from Step 13 and hold them in the vector:

$length1_b(j)$; for $j = 1, \cdots, K2$

```
Step 15: find the start and end position of the founded groups from
            Step 13 and hold them in the matrix:
            $place1_b(j,i)$; for $j = 1,\cdots,K2$; $i = 1,2$
Step 16: set $number1_b = K2$
$A2$ = 33, 56 and 95 for 1024, 2048 and 4096 length of sequence respectively.
    for $j = 1$ to $K2$
        if $length1_b(j) < A2$ then
            $number1_b = number1_b - 1$
            for $i = place1_b(j,1)$ to $place1_b(j,2)$
                $g_{comparison1_b}(i) = 0$
            end
        end
    end
Step 17: set $g_{test1_c} = \left[\tilde{g}(\lfloor N*0.75\rfloor), \tilde{g}(\lfloor N*0.75\rfloor + 1), \cdots, \tilde{g}(N - edge)\right]$.
Step 18: find the index or indexes where ( $g_{test1_c} > 1.01 * Average\ number\ 1$ )
    is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N-edge\rfloor - \lfloor N*0.75\rfloor + 1}$
Step 19: set $g_{comparison1_c}(i_a) = 1$;
            for $a = 1,2,3,\cdots,(\lfloor N-edge\rfloor - \lfloor N*0.75\rfloor + 1)$
            according to the founded indexes of Step 18.
Step 20: find the number of groups in $g_{comparison1_c}$ of consecutive samples
            equal to one and set this number to $K3$.
Step 21: find the length of those groups of consecutive samples equal to one
            from Step 20 and hold them in the vector:
            $length1_c(j)$; for $j = 1,\cdots,K3$
Step 22: find the start and end position of the founded groups from
Step 20 and hold them in the matrix:
            $place1_c(j,i)$; for $j = 1,\cdots,K3$; $i = 1,2$
Step 23: set $number1_c = K3$
        for $j = 1$ to $K3$
            if $length1_c(j) < A1$ then
                $number1_c = number1_c - 1$
                for $i = place1_c(j,1)$ to $place1_c(j,2)$
                    $g_{comparison1_c}(i) = 0$
                end
            end
        end
```

<u>**Comparison to Average number 2:**</u>

```
Step 1: set $g_{average2_a} = \left[\tilde{g}(edge), \tilde{g}(edge+1), \cdots, \tilde{g}(\lfloor N*0.4\rfloor)\right]$
    $g_{average2_b} = \left[\tilde{g}(\lfloor N*0.6\rfloor + 1), \tilde{g}(\lfloor N*0.6\rfloor + 2), \cdots, \tilde{g}(\lfloor N - edge\rfloor)\right]$
Step 2: set $len_{2_a} = \lfloor N*0.4\rfloor - edge$
$len_{2_b} = ((N - edge)) - (\lfloor N*0.6\rfloor)$
```

Step 3: set $Average\ number\ 2 = \left[\frac{g_{average2_a}}{len_{2_a}}, \frac{g_{average2_b}}{len_{2_b}} \right]$

Step 4: set $g_{test2_a} = \left[\tilde{g}\left(\lfloor N*0.24 \rfloor\right), \tilde{g}\left(\lfloor N*0.24 \rfloor + 1\right), \cdots, \tilde{g}\left(\lfloor N*0.35 \rfloor\right) \right]$

Step 5: find the index or indexes where

($g_{test2_a} > 1.002 * Average\ number\ 2(1)$) is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.35 \rfloor - \lfloor N*0.24 \rfloor + 1}$

Step 6: set $g_{comparison2_a}(i_a) = 1$;
for $a = 1, 2, 3, \cdots, \left(\lfloor N*0.35 \rfloor - \lfloor N*0.24 \rfloor + 1\right)$

according to the founded indexes of Step 5.

Step 7: find the number of groups in $g_{comparison2_a}$ of consecutive samples equal to one and set this number to $K4$.

Step 8: find the length of those groups of consecutive samples equal to one from Step 7 and hold them in the vector:

$length2_a(j)$; for $j = 1, \cdots, K4$

Step 9: find the start and end position of the founded groups from Step 7 and hold them in the matrix:

$place2_a(j,i)$; for $j = 1, \cdots, K4$; $i = 1, 2$

Step 10: set $number2_a = K4$

$A3$ = 17, 48 and 129 for 1024, 2048 and 4096 length of sequence respectively.

```
for $j = 1$ to $K4$
    if $length2_a(j) < A3$ then
        $number2_a = number2_a - 1$
        for $i = place2_a(j,1)$ to $place2_a(j,2)$
            $g_{comparison2_a}(i) = 0$
        end
    end
end
```

Step 11: set $g_{test2_b} = \left[\tilde{g}\left(\lfloor N*0.65 \rfloor\right), \tilde{g}\left(\lfloor N*0.65 \rfloor + 1\right), \cdots, \tilde{g}\left(\lfloor N*0.76 \rfloor\right) \right]$

Step 12: find the index or indexes where

($g_{test2_b} > 1.002 * Average\ number\ 2(2)$) is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.76 \rfloor - \lfloor N*0.65 \rfloor + 1}$

Step 13: set $g_{comparison2_b}(i_a) = 1$;
for $a = 1, 2, 3, \cdots, \left(\lfloor N*0.76 \rfloor - \lfloor N*0.65 \rfloor + 1\right)$

according to the founded indexes of Step 12.

Step 14: find the number of groups in $g_{comparison2_b}$ of consecutive samples equal to one and set this number to K5.

Step 15: find the length of those groups of consecutive samples equal to one from Step 14 and hold them in the vector:

$length2_b(j)$; for $j = 1, \cdots, K5$

Step 16: find the start and end position of the founded groups from Step 14 and hold them in the matrix:

$place2_b(j,i)$; for $j = 1, \cdots, K5$; $i = 1, 2$

Step 17: set $number2_b = K5$

```
        for  $j=1$  to $K5$
            if  $length2_b(j) < A3$  then
                $number2_b = number2_b - 1$
                for  $i = place2_b(j,1)$  to  $place2_b(j,2)$
                    $g_{comparison2_b}(i) = 0$
                end
            end
        end
```

Comparison to Average number 3:

Step 1: set $g_{average3_a} = \left[\tilde{g}(edge), \tilde{g}(edge+1), \cdots, \tilde{g}(\lfloor N*0.25 \rfloor)\right]$

$g_{average3_b} = \left[\tilde{g}(\lfloor N*0.25 \rfloor+1), \tilde{g}(\lfloor N*0.25 \rfloor+2), \cdots, \tilde{g}(\lfloor N*0.75 \rfloor)\right]$

$g_{average3_c} = \left[\tilde{g}(\lfloor N*0.75 \rfloor+1), \tilde{g}(\lfloor N*0.75 \rfloor+2), \cdots, \tilde{g}(\lfloor N-edge \rfloor)\right]$

Step 2: set $len_{3_a} = (\lfloor N*0.25 \rfloor) - edge$

$len_{3_b} = (\lfloor N*0.75 \rfloor) - (\lfloor N*0.25 \rfloor)$

$len_{3_c} = ((N-edge)) - (\lfloor N*0.75 \rfloor)$

Step 3: set $Average\ number\ 3 = \left[\frac{g_{average3_a}}{len_{3_a}}, \frac{g_{average3_b}}{len_{3_b}}, \frac{g_{average3_c}}{len_{3_c}}\right]$

Step 4: set $g_{test3_a} = \left[\tilde{g}(\lfloor N*0.34 \rfloor), \tilde{g}(\lfloor N*0.34 \rfloor+1), \cdots, \tilde{g}(\lfloor N*0.45 \rfloor)\right]$

Step 5: find the index or indexes where ($g_{test3_a} > Average\ number\ 3(1)$)

is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.45 \rfloor - \lfloor N*0.34 \rfloor + 1}$

Step 6: set $g_{comparison3_a}(i_a) = 1$;

for $a = 1, 2, 3, \cdots, (\lfloor N*0.45 \rfloor - \lfloor N*0.34 \rfloor + 1)$

according to the founded indexes of Step 5.

Step 7: find the number of groups in $g_{comparison3_a}$ of consecutive samples equal to one and set this number to $K6$.

Step 8: find the length of those groups of consecutive samples equal to one from Step 7 and hold them in the vector:

$length3_a(j)$; for $j = 1, \cdots, K6$

Step 9: find the start and end position of the founded groups from Step 7 and hold them in the matrix:

$place3_a(j,i)$; for $j = 1, \cdots, K6$; $i = 1, 2$

Step 10: set $number3_a = K6$

```
    for  $j=1$  to $K6$
        if  $length3_a(j) < A3$  then
            $number3_a = number3_a - 1$
            for  $i = place3_a(j,1)$  to  $place3_a(j,2)$
                $g_{comparison3_a}(i) = 0$
            end
        end
    end
```

Step 11: set $g_{test3_b} = \left[\tilde{g}(\lfloor N*0.55 \rfloor), \tilde{g}(\lfloor N*0.55 \rfloor+1), \cdots, \tilde{g}(\lfloor N*0.66 \rfloor)\right]$

Step 12: find the index or indexes where ($g_{test3_b} > Average\ number\ 3(3)$) is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.66 \rfloor - \lfloor N*0.55 \rfloor + 1}$

Step 13: set $g_{comparison3_b}(i_a) = 1$;
for $a = 1, 2, 3, \cdots, (\lfloor N*0.66 \rfloor - \lfloor N*0.55 \rfloor + 1)$
according to the founded indexes of Step 12.

Step 14: find the number of groups in $g_{comparison3_b}$ of consecutive samples equal to one and set this number to $K7$.

Step 15: find the length of those groups of consecutive samples equal to one from Step 14 and hold them in the vector:

$$length3_b(j);\ \text{for}\ \ j = 1, \cdots, K7$$

Step 16: find the start and end position of the founded groups from Step 14 and hold them in the matrix:

$$place3_b(j,i);\ \text{for}\ \ j = 1, \cdots, K7;\ \ i = 1, 2$$

Step 17: set $number3_b = K7$

```
    for $j = 1$ to $K7$
        if $length3_b(j) < A3$ then
            $number3_b = number3_b - 1$
            for $i = place3_b(j,1)$ to $place3_b(j,2)$
                $g_{comparison3_b}(i) = 0$
            end
        end
    end
```

Step 18: set $g_{test3_c} = \left[\tilde{g}(\lfloor N*0.45 \rfloor), \tilde{g}(\lfloor N*0.45 \rfloor + 1), \cdots, \tilde{g}(\lfloor N*0.55 \rfloor)\right]$

Step 19: find the index or indexes where
($g_{test3_c} > 1.05 * Average\ number\ 3(2)$)
is true and set those indexes to $i_1, i_2, i_3, \cdots, i_{\lfloor N*0.55 \rfloor - \lfloor N*0.45 \rfloor + 1}$

Step 20: set $g_{comparison3_c}(i_a) = 1$;
for $a = 1, 2, 3, \cdots, (\lfloor N*0.55 \rfloor - \lfloor N*0.45 \rfloor + 1)$
according to the founded indexes of Step 19.

Step 21: find the number of groups in $g_{comparison3_c}$ of consecutive samples equal to one and set this number to $K8$.

Step 22: find the length of those groups of consecutive samples equal to one from Step 21 and hold them in the vector:

$$length3_c(j);\ \text{for}\ \ j = 1, \cdots, K8$$

Step 23: find the start and end position of the founded groups from Step 21 and hold them in the matrix:

$$place3_c(j,i);\ \text{for}\ \ j = 1, \cdots, K8;\ \ i = 1, 2$$

Step 24: *set* $number3_c = K8$
$A4$ = 9, 15 and 63 for 1024, 2048 and 4096 length of sequence respectively.

```
        for $j = 1$ to $K8$
            if $length3_c(j) < A4$ then
                $number3_c = number3_c - 1$
                for $i = place3_c(j,1)$ to $place3_c(j,2)$
            $g_{comparison3_c}(i) = 0$
        end
    end
end
```

Fine tuning on the above obtained results:

The algorithm has also false alarms at the stage of marking the suspected parts as periodic. Therefore, there is a mechanism which deletes some of the false alarms which is described in the following.

```
Step 1: if ( $number1_a == 1$ ) & ( $number1_c > 1$ ) then
        $g_{comparison1_a} = 0$
    end
Step 2: if ( $number1_a > 1$ ) & ( $number1_c == 1$ ) then
        $g_{comparison1_c} = 0$
    end
Step 3: if ( $number2_a == 1$ ) & ( $number2_b > 1$ ) then
        $g_{comparison2_a} = 0$
    end
Step 4: if ( $number2_a > 1$ ) & ( $number2_b == 1$ ) then
        $g_{comparison2_b} = 0$
    end
Step 5: if ( $number3_a > 0$ ) or ( $number3_b > 0$ ) then
        $g_{comparison3_c} = 0$
    end
```

The last stage is to connect all the vectors with the suspicious locations as a periodic part to one outcome vector.

Step 6: Initialize the following vectors with zeros of length *N*:

$g^*_{comparison1_a}$, $g^*_{comparison1_b}$, $g^*_{comparison1_c}$, $g^*_{comparison2_a}$,

$g^*_{comparison2_b}$, $g^*_{comparison3_a}$, $g^*_{comparison3_b}$, $g^*_{comparison3_c}$.

Step 7: Arrange the values in the vectors

$g_{comparison1_a}$, $g_{comparison1_b}$, $g_{comparison1_c}$ $g_{comparison2_a}$,

$g_{comparison2_b}$, $g_{comparison3_a}$, $g_{comparison3_b}$, $g_{comparison3_c}$

to their position in the original vector $\tilde{g}[n]$ by using the vectors from Step 6.

```
$g^*_{comparison1_a}(i + edge) = g_{comparison1_a}(i)$;
                for $i = 1, 2, \cdots, length(g_{comparison1_a})$
$g^*_{comparison1_b}(i + \lfloor N * 0.45 \rfloor) = g_{comparison1_b}(i)$;
                for $i = 1, 2, \cdots, length(g_{comparison1_b})$
$g^*_{comparison1_c}(i + \lfloor N * 0.75 \rfloor) = g_{comparison1_c}(i)$;
                for $i = 1, 2, \cdots, length(g_{comparison1_c})$
$g^*_{comparison2_a}(i + \lfloor N * 0.24 \rfloor) = g_{comparison2_a}(i)$;
```

$$\text{for} \quad i = 1, 2, \cdots, length\left(g_{comparison2_a}\right)$$

$$g^{*}_{comparison2_b}\left(i + \lfloor N * 0.65 \rfloor\right) = g_{comparison2_b}(i);$$

$$\text{for} \quad i = 1, 2, \cdots, length\left(g_{comparison2_b}\right)$$

$$g^{*}_{comparison3_a}\left(i + \lfloor N * 0.34 \rfloor\right) = g_{comparison3_a}(i);$$

$$\text{for} \quad i = 1, 2, \cdots, length\left(g_{comparison3_a}\right)$$

$$g^{*}_{comparison3_b}\left(i + \lfloor N * 0.54 \rfloor\right) = g_{comparison3_b}(i);$$

$$\text{for} \quad i = 1, 2, \cdots, length\left(g_{comparison3_b}\right)$$

$$g^{*}_{comparison3_c}\left(i + \lfloor N * 0.45 \rfloor\right) = g_{comparison3_c}(i);$$

$$\text{for} \quad i = 1, 2, \cdots, length\left(g_{comparison3_c}\right)$$

Step 8: Obtain one output vector which holds the suspicious places in $\tilde{g}[n]$.

$$suspicious\_places[n] = g^{*}_{comparison1_a} + g^{*}_{comparison1_b} + g^{*}_{comparison1_c} + g^{*}_{comparison2_a} + g^{*}_{comparison2_b} + g^{*}_{comparison3_a} + g^{*}_{comparison3_b} + g^{*}_{comparison3_c}.$$

5. Simulation

Wigner Test Vs DFT NIST Test

In this section we first show the simulation results (Table 2) using the Wigner test (the test with the four parameters "Power", "Distance", "Density" and "Highest") for identifying if the tested input sequence is a random sequence or not, compared to those obtained with the DFT test from NIST [18]. For this purpose, eight sequences were applied with different number of cycles, with different cycle length and segment position of the cyclic part in the tested sequence. Each structure was tested with a sequence length of 1024, 2048 and 4096 (please refer to Section III where we have already presented the results for the Wigner test and where the tested sequences were defined). According to Table 2, the DFT test from NIST [18], does not detect in most cases that the non-stationary input sequence is not a random sequence, while the Wigner test (the test with the four parameters "Power", "Distance", "Density" and "Highest") supplies in most cases the right answer. Please note that in Table 2 the red colored values are the indices that classify the sequence as non-random.

New Test Vs NIST Tests

It should be pointed out that the tested input sequences used for generating the results in Table 2 are sequences with a high periodicity within the sequence or have a high length of a repetitive sequence. For this case, the Wigner test (the test with the four parameters "Power", "Distance", "Density" and "Highest") is enough for declaring if the tested input sequence is random or not random. But, for sequences with a low periodicity within the sequence the GGD function [28] [29] is needed. In the following, we denote as the "New test" our new proposed method as described in the previous section and described in Figure 7 containing the Wigner and the GDD functions [27] [28] [29].

Table 3 shows a summary of results obtained by the NIST tests number 1, 2, 3, 6, 7 and 8 [18] and by the "New test" algorithm for low-periodic input sequences (2 periods in a signal) and input sequences having a short periodic segment length (100 - 170 samples of a periodic segment length out of 1024 samples). It can be clearly seen from Table 3, that in most cases the "New test" classifies the incoming sequence as non-random as it should, while the opposite is true for the NIST tests number 1, 2, 3, 6, 7 and 8 [18].

Table 2. Comparison between the DFT test to Wigner.

	Length of sequence	Wigner					DFT	
		Power	Distance	Density	Highest	Random	P value	Random
1	1024	0.9 - 1	0.78	0.51	1208	no	1.2e-04	no
2	1024	0.9 - 1	0.72	0.56	3611	no	0.8364	yes
3	1024	0.8 - 0.9	0.3	0.78	18	no	0.4559	yes
4	1024	0.7 - 0.8	0.25	0.88	6	yes	0.6464	yes
5	1024	0.8 - 0.9	0.39	0.79	19	no	0.4913	yes
6	1024	0.7 - 0.8	0.3	0.9	19	no	0.4913	yes
7	1024	0.9 - 1	0.47	0.83	36	no	0.6881	yes
8	1024	0.9 - 1	0.44	0.76	33	no	0.0665	yes
9	2048	0.9 - 1	0.75	0.6	3282	no	7.2e-12	no
10	2048	0.9 - 1	0.8	0.5	9182	no	0.7151	yes
11	2048	0.7 - 0.8	0.22	0.77	13	no	0.6881	yes
12	2048	0.7 - 0.8	0.26	0.61	7	no	0.1215	yes
13	2048	0.9 - 1	0.29	0.91	48	no	0.207	yes
14	2048	0.7 - 0.8	0.3	0.91	13	yes	0.3304	yes
15	2048	0.9 - 1	0.4	0.84	43	no	0.0027	no
16	2048	0.9 - 1	0.4	0.8	51	no	0.0167	yes
17	4096	0.9 - 1	0.83	0.62	6346	no	1.3e-21	no
18	4096	0.9 - 1	0.7	0.51	24,527	no	0.5281	yes
19	4096	0.7 - 0.8	0.25	0.72	4	no	0.1687	yes
20	4096	0.7 - 0.8	0.3	0.6	11	no	0.0031	no
21	4096	0.9 - 1	0.42	0.92	84	no	0.2758	yes
22	4096	0.7 - 0.8	0.35	0.93	19	no	0.0963	yes
23	4096	0.9 - 1	0.48	0.84	96	no	0.002	no
24	4096	0.9 - 1	0.52	0.81	124	no	0.2175	yes

Table 3. Comparison of NIST test results with the "New test".

Structure	New test	NIST test	Test 1 NIST	Test 2 NIST	Test 3 NIST	Test 6 NIST	Test 7 NIST	Test 8 NIST
Structure number 1	Non random	Non random	0.95	0.518	0.348	0.4913	-	0.0733
	Non random	Random	0.26	0.556	0.144	0.491	0.341	0.364

Continued

	Non random	Random	0.189	0.34	0.231	0.122	0.916	0.837
	Non random	Non random	0.189	0.804	0.537	0.067	-	0.122
	Non random	Random	0.169	0.367	0.799	0.122	0.409	0.102
	Non random	Non random	0.104	0.197	0.047	0.067	-	0.17
	Non random	Random	0.803	0.932	0.575	0.688	0.372	0.793
Structure number 2	Non random	Random	0.532	0.258	0.0541	0.6881	0.553	0.045
	Non random	Non random	0.317	0.709	0.552	0.207	-	0.145
	Non random	Random	0.137	0.403	0.42	0.646	0.538	0.095
	Non random	Random	0.453	0.834	0.629	0.329	0.145	0.865
	Non random	Non random	0.317	0.598	0.247	0.909	-	0.016
Structure number 3	Non random	Random	0.608	0.559	0.163	0.207	0.752	0.736
	Non random	Non random	0.492	0.223	0.912	0.0665	-	0.689
	Non random	Non random	0.95	0.993	0.617	0.908	-	-
	Non random	Random	0.617	0.836	0.497	0.491	0.081	0.865
	Non random	Random	0.617	0.945	0.667	0.456	0.123	0.167
Structure number 4	Non random	Non random	0.708	0.634	0.855	0.207	-	0.736
	Random	Random	0.532	0.879	0.255	0.688	0.458	0.576
	Non random	Random	0.118	0.655	0.193	0.688	0.301	0.322
	Random	Non random	0.851	0.644	-	0.456	-	0.859
	Non random	Non random	0.317	0.056	-	0.1215	-	0.593
Structure number 5	Non random	Random	0.169	0.489	0.418	0.688	0.196	0.767
	Non random	Random	0.211	0.289	0.254	0.491	0.752	0.349
	Non random	Random	0.492	0.984	0.094	0.329	0.659	0.72
	Non random	Random	1	0.997	0.803	0.646	0.538	0.409
	Random	Random	0.901	0.827	0.951	0.909	0.107	0.647
Structure number 6	Non random	Random	0.754	0.308	0.19	0.122	0.752	0.239
	Non random	Random	0.382	0.197	0.144	0.329	0.288	0.249
	Non random	Random	0.249	0.912	0.436	0.688	0.597	0.593

The sequences structure:

Structure number 1: 128, 320, 128, 320, 128.

Structure number 2: 150, 170, 384, 170, 150.

Structure number 3: 100, 824, 100.

Structure number 4: 50, 924, 50.

Structure number 5: 200, 100, 324, 100, 300.

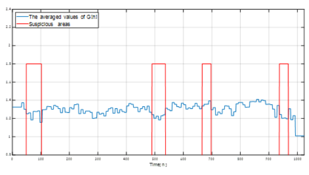
Structure number 6: 312, 100, 200, 100, 312.

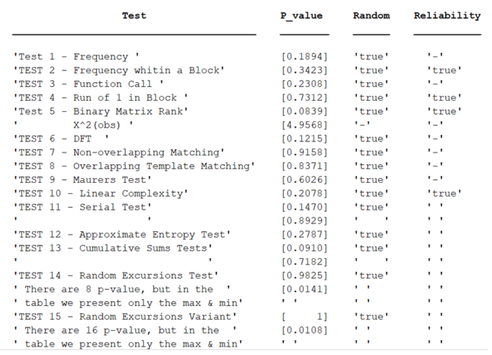
(The number marked in red marks the periodic part in the sequence.)

Simulation results—Random or Non-random:

Simulation results—Estimating the Random places:

Next, we compare the simulated results obtained from our new algorithm "New test" to those obtained by the 15 tests from NIST [18] (Figures 27-30).





```
                    Test                        P_value     Random    Reliability
  ________________________________________     ________    ______    ___________

  'Test 1 - Frequency '                        [0.1894]    'true'      '-'
  'TEST 2 - Frequency whitin a Block'          [0.3423]    'true'      'true'
  'TEST 3 - Function Call '                    [0.2308]    'true'      '-'
  'TEST 4 - Run of 1 in Block '                [0.7312]    'true'      'true'
  'Test 5 - Binary Matrix Rank'                [0.0839]    'true'      'true'
  '         X^2(obs) '                         [4.9568]    '-'         '-'
  'TEST 6 - DFT  '                             [0.1215]    'true'      '-'
  'TEST 7 - Non-overlapping Matching'          [0.9158]    'true'      '-'
  'TEST 8 - Overlapping Template Matching'     [0.8371]    'true'      '-'
  'TEST 9 - Maurers Test'                      [0.6026]    'true'      '-'
  'TEST 10 - Linear Complexity'                [0.2078]    'true'      'true'
  'TEST 11 - Serial Test'                      [0.1470]    'true'      ' '
  '                      '                     [0.8929]    '    '      ' '
  'TEST 12 - Approximate Entropy Test'         [0.2787]    'true'      ' '
  'TEST 13 - Cumulative Sums Tests'            [0.0910]    'true'      ' '
  '                                '           [0.7182]    '    '      ' '
  'TEST 14 - Random Excursions Test'           [0.9825]    'true'      ' '
  ' There are 8 p-value, but in the  '         [0.0141]    ' '         ' '
  ' table we present only the max & min'       ' '         ' '         ' '
  'TEST 15 - Random Excursions Variant'        [     1]    'true'      ' '
  ' There are 16 p-value, but in the  '        [0.0108]    ' '         ' '
  ' table we present only the max & min'       ' '         ' '         ' '
```

structure 1:

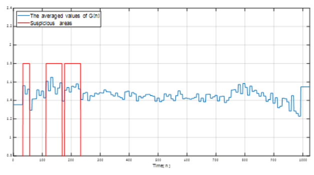
$$y_1(n) = Randomal - 320\ samples$$

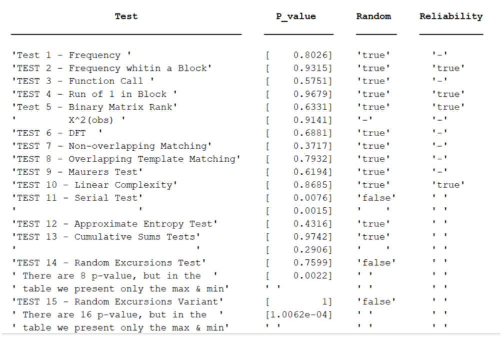
$$y_2(n) = Randomal - 320\ samples$$

$$y_3(n) = Randomal - 128\ samples$$

$$\boldsymbol{m}_1(n) = y_3(n), y_2(n), y_3(n), y_1(n), y_3(n)$$

Figure 27. Result of a sequence from the structure 1, on the left the results from the "New test", on the right the results from the NIST tests.





```
                    Test                        P_value       Random    Reliability
  ________________________________________     __________    ______    ___________

  'Test 1 - Frequency '                        [    0.8026]  'true'      '-'
  'TEST 2 - Frequency whitin a Block'          [    0.9315]  'true'      'true'
  'TEST 3 - Function Call '                    [    0.5751]  'true'      '-'
  'TEST 4 - Run of 1 in Block '                [    0.9679]  'true'      'true'
  'Test 5 - Binary Matrix Rank'                [    0.6331]  'true'      'true'
  '         X^2(obs) '                         [    0.9141]  '-'         '-'
  'TEST 6 - DFT  '                             [    0.6881]  'true'      '-'
  'TEST 7 - Non-overlapping Matching'          [    0.3717]  'true'      '-'
  'TEST 8 - Overlapping Template Matching'     [    0.7932]  'true'      '-'
  'TEST 9 - Maurers Test'                      [    0.6194]  'true'      '-'
  'TEST 10 - Linear Complexity'                [    0.8685]  'true'      'true'
  'TEST 11 - Serial Test'                      [    0.0076]  'false'     ' '
  '                      '                     [    0.0015]  '    '      ' '
  'TEST 12 - Approximate Entropy Test'         [    0.4316]  'true'      ' '
  'TEST 13 - Cumulative Sums Tests'            [    0.9742]  'true'      ' '
  '                                '           [    0.2906]  '    '      ' '
  'TEST 14 - Random Excursions Test'           [    0.7599]  'false'     ' '
  ' There are 8 p-value, but in the  '         [    0.0022]  ' '         ' '
  ' table we present only the max & min'       ' '           ' '         ' '
  'TEST 15 - Random Excursions Variant'        [         1]  'false'     ' '
  ' There are 16 p-value, but in the  '        [1.0062e-04]  ' '         ' '
  ' table we present only the max & min'       ' '           ' '         ' '
```

structure 2:

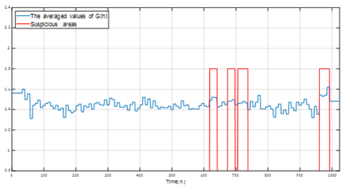
$$y_{1,4}(n) = Randomal - 150\ samples$$

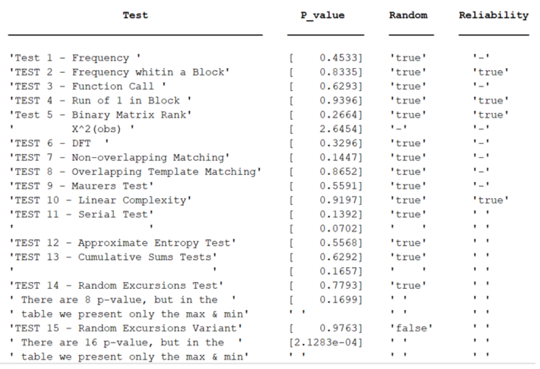
$$y_2(n) = Randomal - 384\ samples$$

$$y_3(n) = Randomal - 170\ samples$$

$$\boldsymbol{m_2(n) = y_1(n), y_3(n), y_2(n), y_3(n), y_4(n)}$$

Figure 28. Result of a sequence from the structure 2, on the left the results from the "New test", on the right the results from the NIST tests.





```
                  Test                        P_value     Random    Reliability
_______________________________________     _________    _______    ___________

'Test 1 - Frequency '                       [  0.4533]   'true'       '-'
'TEST 2 - Frequency whitin a Block'         [  0.8335]   'true'       'true'
'TEST 3 - Function Call '                   [  0.6293]   'true'       '-'
'TEST 4 - Run of 1 in Block '               [  0.9396]   'true'       'true'
'Test 5 - Binary Matrix Rank'               [  0.2664]   'true'       'true'
'         X^2(obs) '                        [  2.6454]   '-'          '-'
'TEST 6 - DFT  '                            [  0.3296]   'true'       '-'
'TEST 7 - Non-overlapping Matching'         [  0.1447]   'true'       '-'
'TEST 8 - Overlapping Template Matching'    [  0.8652]   'true'       '-'
'TEST 9 - Maurers Test'                     [  0.5591]   'true'       '-'
'TEST 10 - Linear Complexity'               [  0.9197]   'true'       'true'
'TEST 11 - Serial Test'                     [  0.1392]   'true'       ' '
'                     '                     [  0.0702]   '    '       ' '
'TEST 12 - Approximate Entropy Test'        [  0.5568]   'true'       ' '
'TEST 13 - Cumulative Sums Tests'           [  0.6292]   'true'       ' '
'                               '           [  0.1657]   '    '       ' '
'TEST 14 - Random Excursions Test'          [  0.7793]   'true'       ' '
' There are 8 p-value, but in the  '        [  0.1699]   ' '          ' '
' table we present only the max & min'      ' '         ' '          ' '
'TEST 15 - Random Excursions Variant'       [  0.9763]   'false'      ' '
' There are 16 p-value, but in the  '       [2.1283e-04] ' '          ' '
' table we present only the max & min'      ' '         ' '          ' '
```

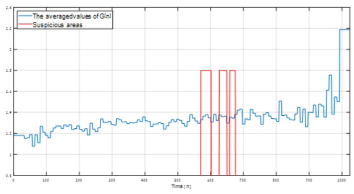
structure 3:

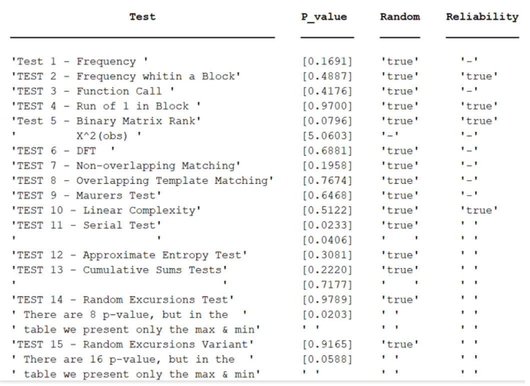
$$y_1(n) = Randomal - 100\ samples$$

$$y_2(n) = Randomal - 824\ samples$$

$$\boldsymbol{m_3(n) = y_1(n), y_2(n), y_1(n)}$$

Figure 29. Result of a sequence from the structure 3, on the left the results from the "New test", on the right the results from the NIST tests.





```
                  Test                        P_value    Random    Reliability
_______________________________________     ________    _______    ___________

'Test 1 - Frequency '                       [0.1691]    'true'       '-'
'TEST 2 - Frequency whitin a Block'         [0.4887]    'true'       'true'
'TEST 3 - Function Call '                   [0.4176]    'true'       '-'
'TEST 4 - Run of 1 in Block '               [0.9700]    'true'       'true'
'Test 5 - Binary Matrix Rank'               [0.0796]    'true'       'true'
'         X^2(obs) '                        [5.0603]    '-'          '-'
'TEST 6 - DFT  '                            [0.6881]    'true'       '-'
'TEST 7 - Non-overlapping Matching'         [0.1958]    'true'       '-'
'TEST 8 - Overlapping Template Matching'    [0.7674]    'true'       '-'
'TEST 9 - Maurers Test'                     [0.6468]    'true'       '-'
'TEST 10 - Linear Complexity'               [0.5122]    'true'       'true'
'TEST 11 - Serial Test'                     [0.0233]    'true'       ' '
'                     '                     [0.0406]    '    '       ' '
'TEST 12 - Approximate Entropy Test'        [0.3081]    'true'       ' '
'TEST 13 - Cumulative Sums Tests'           [0.2220]    'true'       ' '
'                               '           [0.7177]    '    '       ' '
'TEST 14 - Random Excursions Test'          [0.9789]    'true'       ' '
' There are 8 p-value, but in the  '        [0.0203]    ' '          ' '
' table we present only the max & min'      ' '        ' '          ' '
'TEST 15 - Random Excursions Variant'       [0.9165]    'true'       ' '
' There are 16 p-value, but in the  '       [0.0588]    ' '          ' '
' table we present only the max & min'      ' '        ' '          ' '
```

structure 4:

$$y_1(n) = Randomal - 200\ samples$$

$$y_2(n) = Randomal - 324\ samples$$

$$y_3(n) = Randomal - 100\ samples$$

$$y_4(n) = Randomal - 300\ samples$$

$$\boldsymbol{m_4(n) = y_1(n), y_3(n), y_2(n), y_3(n), y_4(n)}$$

Figure 30. Result of a sequence from the structure 4, on the left the results from the "New test", on the right the results from the NIST tests.

According to Figures 27-30, our new proposed algorithm ("New test") finds that all the input sequences are non-random as it should in contrary to the NIST tests [18]. In addition, our new proposed algorithm ("New test") also indicates to those places where repeated samples are found that have already appeared in the tested sequence.

Although the presented examples (Figures 27-30) refer to a sequence length of 1024 samples only, the algorithm can also be applied successfully for longer sequences such as for 2048 or 4096 samples by changing only a few parameters within the algorithm. It should be pointed out that according to [18], the assumption has been made that the size of the sequence length, is large (of the order 10^3 to 10^7) for many of the tests in [18]. For such large sample sizes of the sequence length, asymptotic reference distributions have been derived and applied to carry out the tests [18]. Most of the tests are applicable for smaller values of the sequence length [18]. However, if used for smaller values of the sequence length, the asymptotic reference distributions would be inappropriate and would need to be replaced by exact distributions that would commonly be difficult to compute [18]. As already mentioned earlier in this work, the Wigner function [27] allows us to get the spectrum as a function of time which makes it applicable for the non-stationary input sequence case. The DFT function is not suitable for the non-stationary input sequence case. Different sizes of the input sequence length do not affect the fact that the Wigner function [27] is a proper choice for the non-stationary input sequence. Thus, we just looked for the minimum input sequence length allowed according to [18] in order to have a low computation time.

6. Conclusion

In this paper, we proposed a new approach for estimating the probability of signal randomness degree based on the Wigner and GGD functions which allowed us to classify the input sequence in the time and frequency domains at the same time. Our new proposed approach is suitable also for non-stationary input sequences where the NIST tests fail. In addition, this new proposed algorithm has the ability to indicate on suspicious places of cyclic sections in the tested sequence. Thus, the option to repair or to remove the suspicious places of cyclic sections in the tested input sequence is given with this new approach which was not available until now. Simulation results have confirmed the effectiveness of our new proposed method for estimating the probability of signal randomness degree for non-stationary input sequences. It should be pointed out here that the values for the four parameters ("Power", "Distance", "Density" and "Highest") were not optimized. Thus, it is possible to get even better results in identifying if the tested sequence is random or not.

Funding Statement

The authors would like to thank the Israeli Innovation Authority **and Defender Cyber Technologies LTD** for their support for research under the Academic Knowledge Guidance Program (Nofar) File No. 63907.

Patent Statement

A Patent Application incorporating this paper has been filed.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Martin, H., Peris-Lopez, P., Tapiador, J. E. and San Millan, E. (2016) A New TRNG Based on Coherent Sampling with Self-Timed Rings. *Transactions on Industrial Informatics*, **12**, 91-100. https://doi.org/10.1109/TII.2015.2502183

[2] Inayah, K., Sukmono, B. E., Purwoko, R. and Indarjani, S. (2013) Insertion Attack Effects on Standard PRNGs ANSI X9.17 and ANSI X9.31 Based on Statistical Distance Tests and Entropy Difference Tests. 2013 *International Conference on Computer, Control, Informatics and Its Applications*, Jakarta, 19-21 November 2013, 219-224. https://doi.org/10.1109/IC3INA.2013.6819177

[3] Soorat, R., Madhuri, K. and Vudayagiri, A. (2017) Hardware Random Number Generator for Cryptography. *NANOSYSTEMS: Physics, Chemistry, Mathematics*, **8**, 600-605. https://doi.org/10.17586/2220-8054-2017-8-5-600-605 https://www.researchgate.net/publication/282639432_Hardware_Random_number_Generator_for_cryptography#fullTextFileContent

[4] Mathew, S.K., Srinivasan, S., Anders, M.A., Kaul, H., Hsu, S.K., Sheikh, F., Agarwal, A., Satpathy, S. and Krishnamurthy, R.K. (2012) 2.4 Gbps, 7 mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45 nm CMOS High-Performance Microprocessors. *IEEE Journal of Solid-State Circuits*, **47**, 2807-2821. https://doi.org/10.1109/JSSC.2012.2217631

[5] Goll, M. and Gueron, S. (2018) Randomness Tests in Hostile Environments. *IEEE Transactions on Dependable and Secure Computing*, **15**, 289-294. https://doi.org/10.1109/TDSC.2016.2537799

[6] Petrie, C.S. and Alvin Connelly, J. (1999) The Sampling of Noise for Random Number Generation. *IEEE International Symposium on Circuits and Systems*, Orlando, 30 May-2 June 1999, 26-29. https://doi.org/10.1109/ISCAS.1999.780085

[7] Soucarros, M., Canovas-Dumas, C., Cldire, J., Elbaz-Vincent, P. and Ral, D. (2011) Influence of the Temperature on True Random Number Generators. 2011 *IEEE International Symposium on Hardware-Oriented Security and Trust*, San Diego, 5-6 June 2011, 24-27. https://doi.org/10.1109/HST.2011.5954990

[8] Bahadur, V., Selvakumar Vijendran, D. and Sobha, P.M. (2016) Reconfigurable Side Channel Attack Resistant True Random Number Generator. *International Conference on VLSI Systems, Architectures, Technology and Applications*, Bengaluru, 10-12 January 2016, 1-6. https://doi.org/10.1109/VLSI-SATA.2016.7593048

[9] Prokofiev, A.O., Chirkin A.V. and Bukharov V.A. (2018) Methodology for Quality Evaluation of PRNG, by Investigating Distribution in a Multidimensional Space. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*, Moscow, 29 January-1 February 2018, 355-357. https://doi.org/10.1109/EIConRus.2018.8317105

[10] Prokofiev, A.O. (2019) Development Principles and Classification of PRNG Graphical Tests. *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*, Saint Petersburg and Moscow, 28-31 January 2019, 295-300. https://doi.org/10.1109/EIConRus.2019.8657165

[11] Pareschi, F., Rovatti, R. and Setti, G. (2007) Second-Level NIST Randomness Tests for Improving Test Reliability. *IEEE International Symposium on Circuits and Systems*, New Orleans, 27-30 May 2007, 1437-1440. https://doi.org/10.1109/ISCAS.2007.378572

[12] Márton, K., Homan, M., Suciu, A. and Rasa, I. (2013) The Histogram Test for Randomness Assessment. 2013 *RoEduNet International Conference 12th Edition: Networking in Education and Research*, Iasi, 26-28 September 2013, 1-5.

https://doi.org/10.1109/RoEduNet.2013.6714183

[13] Zhu, S. (2015) A Randomness Test Based on the Distribution of Position for Pre-Specified Pattern. 2015 *International Conference on Computer Science and Mechanical Automation*, Hangzhou, 23-25 October 2015, 166-169. https://doi.org/10.1109/CSMA.2015.40

[14] Fan, Y.T. and Su, G.P. (2014) A New Testing Method of Randomness for True Random Sequences. 2014 *IEEE 5th International Conference on Software Engineering and Service Science*, Beijing, 27-29 June 2014, 537-540. https://doi.org/10.1109/ICSESS.2014.6933624

[15] Fischer, T. (2018) Testing Cryptographically Secure Pseudo Random Number Generators with Artificial Neural Networks. *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*/*12th IEEE International Conference on Big Data Science and Engineering*, New York, 1-3 August 2018, 1214-1223. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00168

[16] Qi, M. and Dong, J. (2009) Research and Application of Entropy in the Sequence Randomness Test. 2009 *Asia-Pacific Conference on Computational Intelligence and Industrial Applications*, Wuhan, 28-29 November 2009, 224-227. https://doi.org/10.1109/PACIIA.2009.5406638

[17] Mrton, K., Bja, V. and Suciu, A. (2014) Parallel Implementation of the Matrix Rank Test for Randomness Assessment. *IEEE 10th International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, 4-6 September 2014, 317-321. https://doi.org/10.1109/ICCP.2014.6937015

[18] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel M., Banks, D., Heckert, A., Dray, J. and San, V. (2010) A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology, Gaithersburg, 2.1-2.40. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[19] Georgescu, C., Simion, E., Nita, A.P. and Toma, A. (2017) A View on NIST Randomness Tests (In)Dependence. *International Conference on Electronics, Computers and Artificial Intelligence*, Targoviste, 29 June-1 July 2017, 1-4. https://doi.org/10.1109/ECAI.2017.8166460

[20] Hoţoleanu, D., Creţ, O., Suciu, A., Gyorfi, T. and Văcariu, L. (2010) Real-Time Testing of True Random Number Generators through Dynamic Reconfiguration. 2010 *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, Lille, 1-3 September 2010, 247-250. https://doi.org/10.1109/DSD.2010.56

[21] Okada, H. and Umeno, K. (2017) Randomness Evaluation with the Discrete Fourier Transform Test Based on Exact Analysis of the Reference Distribution. *IEEE Transactions on Information Forensics and Security*, **12**, 1218-1226. https://doi.org/10.1109/TIFS.2017.2656473

[22] Iwasaki, A. (2020) Deriving the Variance of the Discrete Fourier Transform Test Using Parseval's Theorem. *IEEE Transactions on Information Theory*, **66**, 1164-1170. https://doi.org/10.1109/TIT.2019.2947045

[23] Pareschi, F., Rovatti R. and Setti, G. (2012) On Statistical Tests for Randomness Included in the NIST SP800-22 Test Suite and Based on the Binomial Distribution. *IEEE Transactions on Information Forensics and Security*, **7**, 491-505. https://doi.org/10.1109/TIFS.2012.2185227

[24] Hamano, K. (2005) The Distribution of the Spectrum for the Discrete Fourier Transform Test Included in SP800-22. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E88-A**, 67-73.

https://doi.org/10.1093/ietfec/E88-A.1.67

[25] Zhu, S., Ma, Y., Li, X., Yang, J. and Lin, J. (2020) On the Analysis and Improvement of Min-Entropy Estimation on Time-Varying Data. *IEEE Transactions on Information Forensics and Security*, **15**, 1696-1708. https://doi.org/10.1109/TIFS.2019.2947871

[26] Claasen, T.A.C.M. and Mecklenbrauker, W.F.G. (1980) The Wigner Distribution—A Tool for Time-Frequency Signal Analysis—PART I: Continuous-Time Signals. *Phillips Journal or Research*, **35**, 217-250.

[27] Claasen, T.A.C.M. and Mecklenbrauker, W.F.G. (1980) The Wigner Distribution—A Tool for Time-Frequency Signal Analysis—PART II: Discrete-Time Signals. *Phillips Journal or Research*, **35**, 276-300.

[28] Domínguez-Molina, J.A., González-Farías, G. and Rodríguez-Dagnino, R.M. (2003) A Practical Procedure to Estimate the Shape Parameter in the Generalized Gaussian Distribution. *Universidad de Guanajuato*, *ITESM Campus Monterrey*, Guanajuato, 1-27.

[29] González-Farías, G., Domínguez-Molina, J.A. and Rodríguez-Dagnino, R.M. (2009) Efficiency of the Approximated Shape Parameter Estimator in the Generalized Gaussian Distribution. *IEEE Transactions on Vehicular Technology*, **58**, 4214-4223. https://doi.org/10.1109/TVT.2009.2021270

[30] Marple, L. (1999) Computing the Discrete-Time "Analytic" Signal via FFT. *IEEE Transactions on Signal Processing*, **47**, 2600-2603. https://doi.org/10.1109/78.782222

Appendix

The mathematical description for $\tilde{g}[n]$ and $\hat{g}[n]$ with an input sequence length of 2048 and 4096 respectively is:

$$\tilde{g}[n]=\left[a_0,a_{8q},a_{16q},a_{24q},\cdots,a_{N-8q}\right]\ (\text{time domain})$$

$$\hat{g}[n]=\left[\hat{a}_0,\hat{a}_{8q},\hat{a}_{16q},\hat{a}_{24q},\cdots,\hat{a}_{N-8q}\right]\ (\text{freguency domain})$$

a_j and $\hat{a}_j$ where

$$j=0,8q,16q,\cdots,N-8q\,.$$

For $q=2$:

$$\begin{aligned}a_j=\Bigg[&\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j],\frac{1}{16}\sum_{i=1}^{i=16}G[i+j]\Bigg]\end{aligned}$$

$$\begin{aligned}\hat{a}_j=\Bigg[&\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\\&\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j],\frac{1}{16}\sum_{i=1}^{i=16}\hat{G}[i+j]\Bigg]\end{aligned}$$

For $q=4$:

$$\begin{aligned}a_j=\Bigg[&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\\&\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j],\frac{1}{32}\sum_{i=1}^{i=32}G[i+j]\Bigg]\end{aligned}$$

$$\begin{aligned}\hat{a}_j = \Big[& \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \\ & \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j], \frac{1}{32}\sum\nolimits_{i=1}^{i=32}\hat{G}[i+j] \Big]\end{aligned}$$