INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 2, PP. 443–451
Manuscript received February 21, 2025; revised May, 2025. doi: 10.24425/ijet.2025.153591

Upgrading Frequency Test for Overlapping Vectors and Fill Tree Tests

Krzysztof Mańk

***Abstract*—Randomness testing is one of the essential and easiest tools for evaluating cryptographic primitives. The faster we can test with more refined tests, the greater volume of data that can be reliably tested. This paper we analyze three tests. Starting with a range of observations made for a well-known frequency test for overlapping vectors in binary sequence testing, for which we have obtained precise chi-square statistic computed in $O\left(dt2^{dt}\right)$ instead of $O\left(2^{2dt}\right)$ time, without precomputed tables. Next we focused on two tests from Dieharder: the DAB Fill Tree Test and the DAB Fill Tree 2 Test — for which the probability functions originally were determined empirically. We also draw attention to the errors found in their implementations and the insufficient exploration of the tested sequence in the second test. Even though these tests have been in the package for over 10 years, their significant shortcomings have not been noticed until now.**

***Keywords*—randomness testing, overlapping vectors testing, NIST STS, Dieharder**

I. Introduction

RANDOMNESS testing is widely used in the evaluation of cryptographic primitives by reduction to examine appropriately crafted binary sequences. Obviously, the quality of the analysis increases with the volume of tested data, but the time and cost increase as well.

Dieharder [1] was created thanks to Robert G. Brown, who ported the Diehard package [2] to the C language. The oldest version we found comes from 2007, while the tests to which this work is devoted were included in 2011 and have not been updated since.

We would like to draw attention to the fact that the widespread use of the Dieharder package does not mean its quality. Our attention was drawn to statistically significant differences between the declared significance levels and the observed signaling frequencies obtained when testing large numbers of sequences. Their precise definition and diagnosis of their causes should have been made a long time ago.

While trying to adapt the tree tests to our needs, we reached into their source codes and discovered their shortcomings and limitations:

- for the probability distributions used, only empirical approximations were found, without providing a justification for the adopted sample size,
- data for the Pearson consistency test are incorrectly prepared,
- only one height of a tree is allowed in each test – after recompiling 5 in total,
- only 32-bit words are allowed in the DAB Fill Tree Test,
- rotations by 8 bits of 32-bit words are insufficient to achieve examination of all bits of the sequence.

This work was supported by the grant No. DOB/002/RON/ID1/2018 by The National Centre for Research and Development.

The Author is with the Military University of Technology, Poland (e-mail: krzysztof.mank@wat.edu.pl)

II. Frequency test for overlapping vectors

WE will start by taking a closer look at well-known frequency test for overlapping vectors. One of the first results comes from Good [3]. He proposed using two statistics computed for t and $t-1$ element vectors.

A. Test overview

Supose we have sequence of n d-bit nonoverlapping blocks: $B_1, B_2, ..., B_n$, which we extend adding $t-1$ of the initial elements at the end. Now we create two sequences, the first – of $2 \leq t < n$ element vectors:

$$\begin{aligned}&(B_1, B_2, ..., B_t), (B_2, B_3, ..., B_{t+1}), (B_3, B_4, ..., B_{t+2}), \ldots,\\ &(B_{n-t+1}, B_{n-t+2}, ..., B_n), (B_{n-t+2}, B_{n-t+3}, ..., B_n, B_1),\\ &(B_{n-t+3}, B_{n-t+4}, ..., B_n, B_1, B_2), \ldots,\\ &(B_n, B_1, B_2, ..., B_{t-1}),\end{aligned}$$

and the second of $t-1$ element vectors:

$$\begin{aligned}&(B_1, B_2, ..., B_{t-1}), (B_2, B_3, ..., B_t), (B_3, B_4, ..., B_{t+1}), \ldots,\\ &(B_{n-t+2}, B_{n-t+3}, ..., B_n), (B_{n-t+3}, B_{n-t+4}, ..., B_n, B_1),\\ &(B_{n-t+4}, B_{n-t+5}, ..., B_n, B_1, B_2), \ldots,\\ &(B_n, B_1, B_2, ..., B_{t-2}).\end{aligned}$$

Both sequences consist of n vectors.

Let v_{i_t} for $i = 0..2^{dt}-1$ be numbers of observed occurrences of all possible t elements vectors (dt bit blocks), and $v_{j_{t-1}}$ for $j = 0..2^{d(t-1)}-1$ – numbers of observed occurrences of all possible $t-1$ elements vectors ($d \cdot (t-1)$ bit blocks), where each dt bit block is identified by an integer value, which binary representation this block constitutes.

The test statistic is a simple difference of the two usual Pearson statistics:

$$\psi_t^2 = \frac{2^{dt}}{n} \sum_{i=0}^{2^{dt}-1} v_{i_t}^2 - \frac{2^{d(t-1)}}{n} \sum_{i=0}^{2^{d(t-1)}-1} v_{i_{t-1}}^2,$$

has chi-square distribution with $2^{dt} - 2^{d(t-1)}$ degrees of freedom asymptotically.

Authors of Statistical Test Suite [4] adopted this approach in Serial Test. Two papers published in 2004: [5] and [6] gave means for evaluating exact test statistic. From the first one, we can derive the formula for the covariance matrix of the vector of counts – v_{i_t}, but to get efficient implementations, one needs to store the weak inverse of the covariance matrix. In the second paper, Alhakim proposed a workaround using the matrix's eigenvectors for eigenvalue 1. Alhakim's method is intended for a more general case in which the vector's elements come from any range of natural numbers. This causes the construction of eigenvectors to be unnecessarily complicated, and to obtain equivalent results to using a covariance matrix, it requires repetition of procedure for all vector lengths from 1 up to desired t. That means a lot of computations.

The observations below lead to the exact test statistic with the number of arithmetic calculations similar to Good's approach.

B. Construction of eigenvectors for all eigenvalues

In [5], we can find a formula for test statistic identical to the one received from a quadratic form with the weak inverse of the covariance matrix:

$$S_{d,t} = \frac{1}{n}\sum_{w=1}^{t}\frac{1}{w^2}\sum_{i=1}^{L(w,d,t)}\left(\Psi_i^w \circ v\right)^2,$$

where w are eigenvalues and Ψ_i^w are corresponding eigenvectors, and $L(w,d,t) = \left(2^d-1\right)^{\min\{2,t-w+1\}}\cdot\left(2^d\right)^{\max\{0,t-w-1\}}$ is the number of eigenvectors for eigenvalue w [5]. Alhakim's formula from [6] is a truncation to $w=1$ only, of presented above.

Let $H^m = H^1 \otimes H^{m-1}$, where $H^1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and $\otimes$ denotes Kronecker product, denote Walsh-Hadamard matrix of format $2^m \times 2^m$. By H_i^m for $i = 0..2^m-1$ we will denote rows of the matrix H^m.

For a given pair d and t, we will derive all eigenvectors from the matrix H^{dt}.

There are $L(1,d,t) = \left(2^d-1\right)^t\cdot 2^{d(t-2)}$ eigenvectors for $w=1$, they are:

$$H^{dt}_{i+j\cdot 2^d+2^{d(t-1)}},\ i = 1..2^d-1,\ j = 0..\left(2^d-1\right)2^{d(t-2)}-1.$$

There are $L(2,d,t) = \left(2^d-1\right)^{t-1}\cdot 2^{d(t-3)}$ eigenvectors for $w=2$ and they are:

$$\frac{1}{\sqrt{2}}\left(H^{dt}_{i+j\cdot 2^d+2^{d(t-2)}} + H^{dt}_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^d}\right),$$
$$i = 1..2^d-1,\ j = 0..\left(2^d-1\right)2^{d(t-3)}-1.$$

For any $w<t$, its eigenvectors are given by:

$$\frac{1}{\sqrt{w}}\sum_{k=1}^{w} H^{dt}_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^{d(k-1)}},$$
$$i = 1..2^d-1,\ j = 0..\left(2^d-1\right)2^{d(t-w-1)}-1,$$

and for $w=t$ we have:

$$\frac{1}{\sqrt{t}}\sum_{k=1}^{t} H^{dt}_{i2^{d(k-1)}},\ i = 1..2^d-1.$$

Due to the linearity of the dot product, we can rewrite the formula for the test statistic:

$$S_{d,t} = \frac{1}{n}\sum_{w=1}^{t-1}\frac{1}{w^2}\sum_{i=1}^{2^d-1}\sum_{j=0}^{\left(2^d-1\right)2^{d(t-w-1)}-1}$$
$$\left(\sum_{k=0}^{w-1}\left(H^{dt}_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^{dk}}\circ v\right)\right)^2 +$$
$$+\frac{1}{n}\frac{1}{t^2}\sum_{i=1}^{2^d-1}\left(\sum_{k=0}^{t-1}\left(H^{dt}_{i2^{dk}}\circ v\right)\right)^2.$$

Because all dot products above constitute elements of the Walsh-Hadamard transform of vector v, which we will denote as V, we finally get:

$$S_{d,t} = \frac{1}{n}\sum_{w=1}^{t-1}\frac{1}{w^2}\sum_{i=1}^{2^d-1}\sum_{j=0}^{\left(2^d-1\right)2^{d(t-w-1)}-1}$$
$$\left(\sum_{k=0}^{w-1} V_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^{dk}}\right)^2 +$$
$$+\frac{1}{n}\frac{1}{t^2}\sum_{i=1}^{2^d-1}\left(\sum_{k=0}^{t-1} V_{i2^{dk}}\right)^2.$$

Vector V should be computed using the fast Walsh-Hadamard transform, whose time complexity is $O\left(dt2^{dt}\right)$.

C. Structure of the count vector and its utilization

Since consecutive observed vectors $(B_i, B_{i+1}, ..., B_{i+t-1})$ overlap on $t-1$ blocks and we have extended examined sequence by as many its initial blocks, then for every possible $d(t-1)$ bit block value we can write an equation:

$$\sum_{k=0}^{2^d-1} v_{k\cdot 2^{d(t-1)}+i} = \sum_{k=0}^{2^d-1} v_{k+i\cdot 2^{d(t-1)}},\ i = 0..2^{d(t-1)}-1.$$

The additional equation is obvious:

$$\sum_{k=0}^{2^{dt}-1} v_k = n.$$

This system of equations has order $2^{d(t-1)}$ thus allows to determine $2^{d(t-1)}$ of the elements of the vector v as a linear combination of n and the rest of them, that is $2^{dt}-2^{d(t-1)}$, which is consistent with the stated number of degrees of freedom of the test statistic $S_{d,t}$.

Walsh-Hadamard transform of modified this way vector v has an interesting property – elements $V_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^{dk}}$ for a given trio (w,i,j) and every $k = 0..w-1$ are equal. The same applies to $V_{i2^{dk}}$, of course. This leads to further simplification of the test statistic:

$$S_{d,t} = \frac{1}{n}\sum_{w=1}^{t-1}\sum_{i=1}^{2^d-1}\sum_{j=0}^{\left(2^d-1\right)2^{d(t-w-1)}-1}$$

$$\left(V_{\left(i+j\cdot 2^d+2^{d(t-2)}\right)2^{d(w-1)}}\right)^2+\frac{1}{n}\sum_{i=1}^{2^d-1}\left(V_{i2^{d(t-1)}}\right)^2.$$

In a such setup, initial $2^{d(t-1)}$ elements of V are obsolete.

III. Fill Tree Test for a binary sequence

The basis of the test is the procedure of randomly walking through a binary tree of height h. In each iteration, the tree is emptied. Then, subsequent bits of the sequence are interpreted as an indicator of delving into the left or right sub-tree until an unvisited node or leaf is found, after which it returns to the root. The iteration ends when a leaf that has already been visited is reached, which we will call collision. Two test statistics are determined:

- number of unique nodes or leaves visited – the observed distribution is compared using the Pearson test with the theoretical distribution,
- position of the leaf for which the collision occurred – the observed distribution is compared using the Pearson test with a uniform distribution.

A. Implementation in Dieharder

Instead of a theoretical distribution of the number of unique nodes visited until the collision occurred, it was established empirically based on $6\cdot 10^9$ samples. In the implementation, trees of height seven are used, and the lowest used probability is $3.33333333333\cdot 10^{-10}$. From the central limit theorem, it is easy to calculate that there is a 48% probability of it being off by at least 50% and 16% – by 100%. [7]

Even in more realistic scenarios, we have obtained:

- 1.5% probability of 1% or more deviation of the lowest used probability for 10 MB sequence (default in DIEHARD),
- 19.6% probability of 2% or more deviation of the lowest used probability for 1 Gib sequence,
- 52% probability of 1% or more deviation of the lowest used probability for 1 Gib sequence.

Below, we present our approach to the analytic evaluation of the probability function for the number of unique nodes or leaves visited before collision.

The following fragment can be found in the implementation:

```
/* Calculate expected counts. */
for (i = 0; i < target; i++) {
  expected[i] = targetData[i] *
test[0]->tsamples;
  if (expected[i] < 4) {
    if (end == 0) start = i;
  } else if (expected[i] > 4) end = i;
}
```

Its function, in addition to calculating the expected values of the number of observations for each number of visited nodes, is also to reject those with a value less than 4. However, there is no grouping here, only a truncation of the arrays range, which results that the vectors of observations and expected values transferred to the function implementing the Pearson's consistency test have different sums, which is unacceptable in this test.

B. Transition probability matrix approach

Under the assumption that the tested sequence is a realization of independent uniformly distributed binary variables, we can swap sub-trees at any level of the tree. Combining them into classes significantly reduces the number of cases considered.

We will start from a tree of height 2. Figure 1 shows the state graph of a tree while it is being filled. We marked the states corresponding to the collision that occurred when k nodes (including the root) were visited with squares. These are absorbing states.

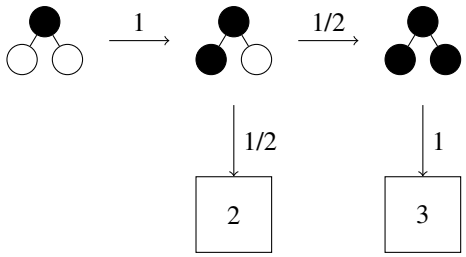

Fig. 1. State graph for a tree of height 2

The following matrix of transition probabilities corresponds to this graph:

$$M_2=\begin{bmatrix}0&1&0&0&0\\0&0&\frac{1}{2}&\frac{1}{2}&0\\0&0&0&0&1\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}.$$

While the longest path in this graph is of length 3, it is sufficient to calculate the third power of the matrix M_2, and from its first row, we would read the desired probabilities. We have:

$$M_2^3=\begin{bmatrix}0&0&0&\frac{1}{2}&\frac{1}{2}\\0&0&0&\frac{1}{2}&\frac{1}{2}\\0&0&0&0&1\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}.$$

We can do the same for a tree of height 3. The transition graph and matrix are on the figures 2 and 3.

From 7-th power of M_3 we get vector of probabilities:

$$\frac{1}{8},\frac{1}{4},\frac{5}{16},\frac{15}{64},\frac{5}{64}.$$

Obtained for a tree of height four matrix is of format 76×76, so we present only obtained probabilities:

$$\frac{1}{2^6},\frac{3}{2^6},\frac{45}{2^9},\frac{535}{2^{12}},\frac{1335}{2^{13}},\frac{355}{2^{11}},\frac{5115}{2^{15}},$$
$$\frac{30525}{2^{18}},\frac{9075}{2^{17}},\frac{32175}{2^{20}},\frac{75075}{2^{23}},\frac{10725}{2^{23}}.$$

For height 7, implemented in Dieharder, the matrix dimension is only slightly less than $3.4\cdot 10^{12}$, so we need a much simpler method.

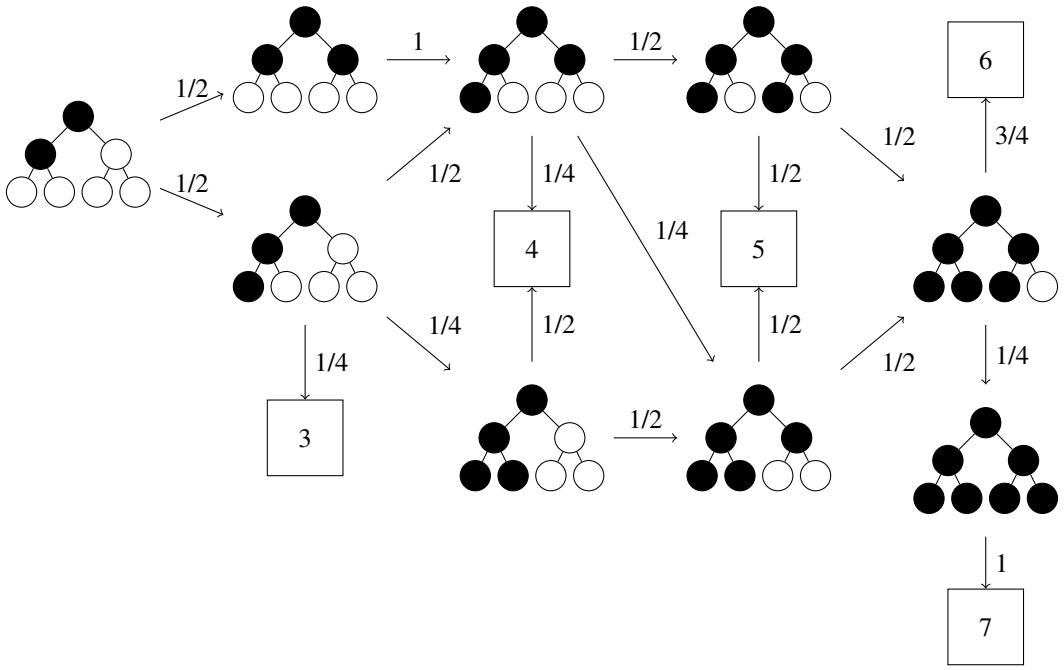

Fig. 2. State graph for a tree of height 3

$$M_3 = \begin{bmatrix}
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{3}{4} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

Fig. 3. State transition matrix for a tree of height 3

TABLE I
PROBABILITIES $p_{h,k}$ AND $s_{h,k}$ FOR HEIGHT 2

k	$p_{2,k}$	$s_{2,k}$
0	0	1
1	0	1
2	1/2	1/2
3	1/2	0

C. Iterative approach

We will consider two sets of connected probabilities:

- $p_{h,k}$ – of collision occurring when $k = 0..2^h - 1$ nodes were visited,
- $s_{h,k}$ – of collision not occurring when $k = 0..2^h - 1$ nodes were visited,

where h is tree height and

$$s_{h,k} = 1 - \frac{p_{h,k}}{\sum_{m=k}^{2^h-1} p_{h,m}}.$$

Our starting point will be probabilities computed for a tree of height 2, probabilities $p_{h,k}$ and $s_{h,k}$ were gathered in the table I.

The calculation for a tree of height 3, we will start from „the last save" configuration, e.t. the highest number of visited nodes without the possibility of collision: 3 – 1 as root and 2 in sub-trees. There are three possible ways of distributing these two elements in the sub-trees. Their probabilities can be determined using the binomial distribution with the number of trials two, and the probability of success is $1/2$.

Intermediate probabilities we will denote as $q_{h,l,r}$, where l is the number of visited nodes in the left sub-tree and r – in the right one.

We begin with probabilities:

$$q_{3,0,2} = \binom{2}{0} \cdot 2^{-2} = \frac{1}{4},\ q_{3,1,1} = \binom{2}{1} \cdot 2^{-2} = \frac{1}{2},$$
$$q_{3,2,0} = \binom{2}{2} \cdot 2^{-2} = \frac{1}{4},$$

and will update them according to the formula:

$$q_{h,l,r} = \frac{1}{2}\left(q_{h,l-1,r} \cdot s_{h-1,l-1} + q_{h,l,r-1} \cdot s_{h-1,r-1}\right),$$

until the $q_{h,2^{h-1}-1,2^{h-1}-1}$ is reached, so we get:

$$q_{3,0,3} = \frac{1}{2} q_{3,0,2} \cdot s_{2,2} = \frac{1}{16},$$
$$q_{3,1,2} = \frac{1}{2}\left(q_{3,0,2} \cdot s_{2,0} + q_{3,1,1} \cdot s_{2,1}\right) = \frac{3}{8},$$
$$q_{3,2,1} = \frac{1}{2}\left(q_{3,1,1} \cdot s_{2,1} + q_{3,0,2} \cdot s_{2,0}\right) = \frac{3}{8},$$
$$q_{3,3,0} = \frac{1}{2} q_{3,2,0} \cdot s_{2,2} = \frac{1}{16}.$$

The next steps are slightly different, which results from the saturation of the capacity of individual subtrees:

$$q_{3,1,3} = \frac{1}{2}\left(q_{3,0,3}\cdot s_{2,0} + q_{3,1,2}\cdot s_{2,2}\right) = \frac{1}{8},$$
$$q_{3,2,2} = \frac{1}{2}\left(q_{3,1,2}\cdot s_{2,1} + q_{3,2,1}\cdot s_{2,1}\right) = \frac{3}{8},$$
$$q_{3,3,1} = \frac{1}{2}\left(q_{3,3,0}\cdot s_{2,0} + q_{3,2,1}\cdot s_{2,2}\right) = \frac{1}{8},$$
$$q_{3,2,3} = \frac{1}{2}\left(q_{3,1,3}\cdot s_{2,1} + q_{3,2,2}\cdot s_{2,2}\right) = \frac{5}{32},$$
$$q_{3,3,2} = \frac{1}{2}\left(q_{3,3,1}\cdot s_{2,1} + q_{3,2,2}\cdot s_{2,2}\right) = \frac{5}{32},$$
$$q_{3,3,3} = \frac{1}{2}\left(q_{3,2,3}\cdot s_{2,2} + q_{3,3,2}\cdot s_{2,2}\right) = \frac{5}{64}.$$

From Kolmogorov's axiomatic definition of probability, we know that the above probabilities, in layers, should sum to 1, but they do not. Fortunately, this only means that we have not included all events—collisions.

For three nodes visited:

$$p_{3,3} = 1 - q_{3,0,3} - q_{3,1,2} - q_{3,2,1} - q_{3,3,0} = \frac{1}{8},$$

for four nodes visited:

$$p_{3,4} = 1 - q_{3,1,3} - q_{3,2,2} - q_{3,3,1} - p_{3,3} = \frac{1}{4},$$

for five nodes visited:

$$p_{3,5} = 1 - q_{3,2,3} - q_{3,3,2} - p_{3,3} - p_{3,4} = \frac{5}{16},$$

for six nodes visited:

$$p_{3,6} = 1 - q_{3,3,3} - p_{3,3} - p_{3,4} - p_{3,5} = \frac{15}{64},$$

and finally:

$$p_{3,7} = 1 - p_{3,3} - p_{3,4} - p_{3,5} - p_{3,6} = q_{3,3,3} = \frac{5}{64}.$$

The above calculations can be summarized in a general formula:

$$p_{h,k} = \frac{1}{2}\sum_{l=\max\{h-1,k-2^{h-1}\}}^{\min\{k-1,2^{h-1}-1\}} q_{h,l,k-l-1}\cdot\left(1 - s_{h-1,l}\right).$$

The obtained values are in perfect agreement with those obtained in III-B from M_3^7. This is no coincidence; if we carefully examine the significant products performed when calculating the 7-th power of M_3, it would turn out that they are exactly the same as above.

D. Comparison with the Dieharder

Referring to the test implementation in Dieharder, we obtained a probability of more than 50% error in determining the lowest tabulated probability equal 48%. Using the discussed method, we obtained $1.69627368106\cdot 10^{-10}$ instead of $3.33333333333\cdot 10^{-10}$, which means an error of 49% with respect to the original value and 97% with respect to the correct one.

In the set of probabilities exceeding $6.9\cdot 10^{-7}$, which corresponds to the expected value of the number of observations equal 4, which is used in the implementation, the maximum relative error slightly exceeds 1%.

The impact of the discussed discrepancies on p-value obtained in Pearson's consistency test is negligible.

IV. Fill Tree Test for a sequence of bit-blocks

The Fill Tree Test case is more complicated than the previous one, and we will complicate it even further.

Suppose that tested (P)RNG produces sequences of d-bit blocks interpreted as a natural numbers from 0 to $2^d - 1$. We repeatedly try to place them in a binary tree of height h. We start from an empty tree, so the first block is placed as a root, and for every next block, we search for the empty node or leaf, choosing the left sub-tree if the current number is less than one in the current node or the right sub-tree otherwise. We finish the iteration when a leaf is reached, and it is already occupied, which we will call a collision. The test consists of chi-square statistics – the first one examines the compliance of the distribution of leaf positions on which collisions occurred, and the second one is the distribution of the number of elements placed in the tree before the collision.

However, knowledge of the theoretical distribution is necessary for the compliance test, which we provide below.

A. Height 2

Our starting point will be a tree of height 2, filled with numbers from the interval $\{A, ..., B-1\}$. We have four ways in which a given iteration can end, and we will calculate the probabilities of each of them occurring:

- collision on the left leaf $(L = 0)$ with two elements $(E = 2)$ in the tree:

$$P_2\left(L = 0 \wedge E = 2\right)(A,B) = \sum_{a=A}^{B-1}\sum_{b=A}^{a-1}\sum_{c=A}^{a-1}\frac{1}{(B-A)^3} = \\ = \frac{\left(1 + \left(2\left(B-A\right) - 3\right)\left(B-A\right)\right)}{6\left(B-A\right)^2},$$

- collision on the right leaf $(L = 1)$ with two elements $(E = 2)$ in the tree:

$$P_2\left(L = 1 \wedge E = 2\right)(A,B) = \sum_{a=A}^{B-1}\sum_{b=a}^{B-1}\sum_{c=a}^{B-1}\frac{1}{(B-A)^3} = \\ = \frac{\left(1 + \left(2\left(B-A\right) + 3\right)\left(B-A\right)\right)}{6\left(B-A\right)^2},$$

- collision on the left leaf $(L = 0)$ with three elements $(E = 3)$ in the tree:

$$P_2\left(L = 0 \wedge E = 3\right)(A,B) = \\ = 2\sum_{a=A}^{B-1}\sum_{b=A}^{a-1}\sum_{c=a}^{B-1}\sum_{e=A}^{a-1}\frac{1}{(B-A)^4} = \frac{(B-A)^2 - 1}{6\left(B-A\right)^2},$$

- collision on the right leaf $(L = 1)$ with three elements $(E = 3)$ in the tree:

$$P_2\left(L = 1 \wedge E = 3\right)(A,B) = \\ = 2\sum_{a=A}^{B-1}\sum_{b=A}^{a-1}\sum_{c=a}^{B-1}\sum_{e=a}^{B-1}\frac{1}{(B-A)^4} = \frac{(B-A)^2 - 1}{6\left(B-A\right)^2}.$$

Below, we will denote these probabilities briefly as: $P_{2,0,2}(A,B)$, $P_{2,1,2}(A,B)$, $P_{2,0,3}(A,B)$, $P_{2,1,3}(A,B)$.

Because we will use a method similar to the one in the previous case, we will need one more set of probabilities – of successfully placing k elements in a tree:

- for $k=0$: $S_{2,0}(A,B)=1$,
- for $k=1$: $S_{2,1}(A,B)=1$,
- for $k=2$: $S_{2,2}(A,B)=1$,
- for $k=3$:

$$S_{2,3}(A,B)=1-P_{2,0,2}(A,B)-P_{2,1,2}(A,B)=$$
$$=P_{2,0,3}(A,B)+P_{2,1,3}(A,B)=\frac{(B-A)^2-1}{3(B-A)^2}.$$

These formulas will be useful later. Now, taking $A=0$ and $B=2^d$, we obtain the desired distributions for a tree of height 2.

Probability of collision occurring:

- on the left leaf:

$$P_{2,0,*}\left(0,2^d\right)=P_{2,0,2}\left(0,2^d\right)+P_{2,0,3}\left(0,2^d\right)=$$
$$=\frac{2^d-1}{2^{d+1}},$$

- on the right leaf:

$$P_{2,1,*}\left(0,2^d\right)=P_{2,1,2}\left(0,2^d\right)+P_{2,1,3}\left(0,2^d\right)=$$
$$=\frac{2^d+1}{2^{d+1}},$$

- when none of elements were placed: $P_{2,*,0}\left(0,2^d\right)=0$,
- and 1 element was placed: $P_{2,*,1}\left(0,2^d\right)=0$,
- and two elements were placed:

$$P_{2,*,2}\left(0,2^d\right)=P_{2,0,2}\left(0,2^d\right)+P_{2,1,2}\left(0,2^d\right)=$$
$$=\frac{2^{2d+1}+1}{3\cdot 2^{2d}},$$

- and three elements were placed:

$$P_{2,*,3}\left(0,2^d\right)=P_{2,0,3}\left(0,2^d\right)+P_{2,1,3}\left(0,2^d\right)=$$
$$=\frac{2^{2d}-1}{3\cdot 2^{2d}}.$$

B. Height 3

For the height three things become more complex or, to be more precise, we have definitely more cases.

To allow for a collision for the next element, at least three elements have to be placed in a tree of height 3, so we skip insignificant cases in the formulas below. Probabilities for collision occurring on a given leaf can be derived as follows:

$$P_{3,0,*}(A,B)=\sum_{a=A}^{B-1}\frac{1}{B-A}\left(\sum_{k=3}^{6}\left(P_{2,0,2}(A,a)\,P_{2,*,k-3}(a,B)\cdot\right.\right.$$
$$\cdot\left(\frac{a-A}{B-A}\right)^3\left(\frac{B-a}{B-A}\right)^{k-3}\binom{k-1}{2}+$$
$$+P_{2,0,3}(A,a)\,P_{2,*,k-3}(a,B)\cdot$$
$$\left.\left.\cdot\left(\frac{a-A}{B-A}\right)^4\left(\frac{B-a}{B-A}\right)^{k-3}\binom{k}{3}\right)\right)=$$
$$=\frac{(B-A-1)(B-A-2)}{360(B-A)^5}\cdot$$
$$\cdot\left(90(B-A)^3+29(B-A)^2-3(B-A)-2\right),$$

$$P_{3,1,*}(A,B)=\sum_{a=A}^{B-1}\frac{1}{B-A}\left(\sum_{k=3}^{6}\left(P_{2,1,2}(A,a)\,P_{2,*,k-3}(a,B)\cdot\right.\right.$$
$$\cdot\left(\frac{a-A}{B-A}\right)^3\left(\frac{B-a}{B-A}\right)^{k-3}\binom{k-1}{2}+$$
$$+P_{2,1,3}(A,a)\,P_{2,*,k-3}(a,B)\cdot$$
$$\left.\left.\cdot\left(\frac{a-A}{B-A}\right)^4\left(\frac{B-a}{B-A}\right)^{k-3}\binom{k}{3}\right)\right)=$$
$$=\frac{(B-A-1)(B-A+1)}{360(B-A)^5}\cdot$$
$$\cdot(2(B-A)+1)(5(B-A)+2)(9(B-A)-2),$$

similarly we get:

$$P_{3,2,*}(A,B)=\frac{(B-A-1)(B-A+1)}{360(B-A)^5}\cdot$$
$$\cdot(2(B-A)-1)(5(B-A)-2)(9(B-A)+2),$$
$$P_{3,3,*}(A,B)=\frac{(B-A+1)(B-A+2)}{360(B-A)^5}\cdot$$
$$\cdot\left(90(B-A)^3-29(B-A)^2-3(B-A)+2\right),$$

and probabilities for collision occurring and:

- 0, 1, 2 elements were placed: $P_{3,*,k}(A,B)=0$, $k\in\{0,1,2\}$,
- 3 elements were placed:

$$P_{3,*,3}(A,B)=\sum_{a=A}^{B-1}\frac{1}{B-A}\left(P_{2,*,2}(A,a)\left(\frac{a-A}{B-A}\right)^3+\right.$$
$$\left.+P_{2,*,2}(a,B)\cdot\left(\frac{B-a}{B-A}\right)^3\right)=\frac{(B-A)^2+2}{3(B-A)^2},$$

- 4 elements were placed:

$$P_{3,*,4}(A,B)=\sum_{a=A}^{B-1}\frac{1}{B-A}\left(P_{2,*,2}(A,a)\cdot\right.$$
$$\cdot\left(\frac{a-A}{B-A}\right)^3\left(\frac{B-a}{B-A}\right)\binom{3}{2}+$$
$$+P_{2,*,2}(a,B)\left(\frac{B-a}{B-A}\right)^3\left(\frac{a-A}{B-A}\right)\binom{3}{2}+$$
$$+P_{2,*,3}(A,a)\left(\frac{a-A}{B-A}\right)^4+$$
$$\left.+P_{2,*,3}(a,B)\left(\frac{B-a}{B-A}\right)^4\right)=$$
$$=\frac{(B-A)^4-1}{3(B-A)^4},$$

- 5 elements were placed:

$$P_{3,*,5}(A,B) = \\ = \frac{(B-A-1)(B-A+1)\left(2(B-A)^2+1\right)}{9(B-A)^4},$$

- 6 elements were placed:

$$P_{3,*,6}(A,B) = \frac{(B-A-1)(B-A+1)}{21(B-A)^6}\cdot \\ \cdot(B-A-2)(B-A+2)\left(2(B-A)^2+3\right),$$

- 7 elements were placed:

$$P_{3,*,7}(A,B) = \frac{(B-A-1)(B-A+1)}{63(B-A)^6}\cdot \\ \cdot(B-A-2)(B-A+2)(B-A-3)(B-A+3).$$

After observing the formulas above, we concluded that we only need to iterate one group of probabilities – of a collision occurring on the l-th leaf when k elements were already placed: $P_{h,l,k}(A,B)$. So, we can move on to generalizations.

Distributions used in the test can be obtained from the formulas above by simply replacing $B-A$ by 2^d in the final formulas.

C. Generalization

As mentioned above, the only probability which we truly need to iterate is $P_{h,l,k}(A,B)$ – probability of collision occurring on the l-th leaf of a tree of height h, when k elements where already placed.

For leafs $l \in \left\{0,...,2^{h-2}-1\right\}$ we have:

$$P_{h,l,k}(A,B) = \sum_{a=A}^{B-1}\frac{1}{B-A}\sum_{m=\max\{h-1,k-2^{h-1}\}}^{\min\left\{k-1,2^{h-1}-1\right\}} \\ P_{h-1,l,m}(A,a)\,S_{h-1,k-m-1}(a,B)\cdot \\ \cdot\frac{(a-A)^{m+1}(B-a)^{k-m-1}}{(B-A)^k}\binom{k-1}{m},$$

and for leafs $l+2^{d-2} \in \left\{2^{h-2},...,2^{h-1}-1\right\}$:

$$P_{h,l+2^{d-2},k}(A,B) = \sum_{a=A}^{B-1}\frac{1}{B-A}\sum_{m=\max\{h-1,k-2^{h-1}\}}^{\min\left\{k-1,2^{h-1}-1\right\}} \\ P_{h-1,l,m}(a,B)\,S_{h-1,k-m-1}(A,a)\cdot \\ \cdot\frac{(a-A)^{k-m-1}(B-a)^{m+1}}{(B-A)^k}\binom{k-1}{m}.$$

From it, we can derive any other:

- probability of a collision occurring on a given leaf:

$$P_{h,l,*}(A,B) = \sum_{k=h}^{2^h-1}P_{h,l,k}(A,B),$$

- probability of collision occurring when k elements where already placed:

$$P_{h,*,k}(A,B) = \sum_{l=0}^{2^{h-1}-1}P_{h,l,k}(A,B),$$

- probability of successful placing of k elements into a tree:

$$S_{h,k}(A,B) = \sum_{m=k}^{2^h-1}P_{h,*,m}(A,B).$$

The calculations are easy but time-consuming, we used Maple in our work.

D. Comparison with the Dieharder

In Dieharder, one can find this test implemented as *DAB Fill Tree Test*, but it operated only on 32-bit blocks and trees of height 4, although you can find the table of probabilities for height 5 in the code as well.

The 32-bit block is so large that it is possible to assume the continuous case in analytic computation, which makes them a bit simpler. What's more, the probabilities of collision occurring on any leaf are all the same. The same can be obtained from our results by determining the limit as d approaches infinity.

Both are placed in the code tables of probabilities and appear to be estimated empirically, maybe with some "minded" corrections for the case of height 4. All values are, at least, close enough to be obtained by us. The impact of their inaccuracy on Pearson's statistics' obtained values seems negligible.

Values based on our work are presented below, but we strongly recommend reading the next paragraph before using them.

For a tree of height $h = 4$ and $d = 32$ probabilities are (for $k = 0..15$):

```
0.0, 0.0, 0.0, 0.0, 0.13333333,
0.20000000, 0.20634921, 0.17857143,
0.13007055, 0.08183422, 0.04338624,
0.01851852, 0.00617284, 0.00151172,
0.00023516, 0.00001680,
```

and for the continuous case:

$$0,0,0,0,\frac{2}{15},\frac{1}{5},\frac{13}{63},\frac{5}{28},\frac{295}{2268},\frac{232}{2835}, \\ \frac{41}{945},\frac{1}{54},\frac{1}{162},\frac{2}{1323},\frac{2}{8505},\frac{1}{59535}.$$

For a tree of height $h = 5$ and $d = 32$ probabilities are (for $k = 0..31$):

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.04444444,
0.08888889, 0.11825397, 0.13165785,
0.13134039, 0.12072310, 0.10338918,
0.08302243, 0.06274687, 0.04466924,
0.02989331, 0.01873895, 0.01096132,
0.00595993, 0.00299902, 0.00138902,
0.00058795, 0.00022540, 0.00007739,
0.00002348, 0.00000618, 0.00000138,
0.00000025, 0.00000004, 0.00000000,
0.00000000, 0.00000000.
```

Of course, the last three values are not equal to 0, they are: $3.9498747\cdot10^{-9}$, $2.7303281\cdot10^{-10}$, and $9.1010938\cdot10^{-12}$.

E. What is wrong with the DAB Fill Tree Test?

After reading the previous paragraph, the reader may have concluded that the test has been used correctly so far, and our calculations do not add anything significant, which would not be largely wrong.

Of course, in the implementation of this test, the error related to grouping too few classes, which we described in point III-A, was repeated. However, there is something more.

In the test description we find information, that words from the RNG are rotated, and from the implementation we can deduce that this is cyclic rotation by 8, 16 and 24 bits. It is said that it is for better detection of RNGs that are biased on high, middle, or low bytes.

In our opinion, without such rotation, the test has almost no capability of detecting any bias on bits other than the 4-th significant when using trees of height 4.

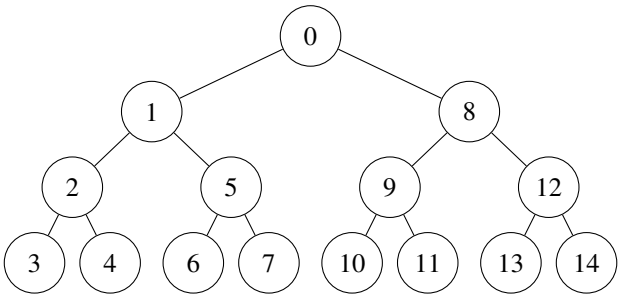

Fig. 4. Tree filling order

In the picture 4, we present a tree of height 4 with nodes labeled according to the filling order that guarantees the maximum number of comparisons for a given tree filling. For example, there are eight ways to fill a tree with four elements, which can lead to a collision when trying to add a fifth element – filling nodes:

1) 0, 1, 2, 3,
2) 0, 1, 2, 4,
3) 0, 1, 5, 6,

...

7) 0, 8, 12, 13,
8) 0, 8, 12, 14.

But they all lead to the same number of comparisons – 9:

- none when placing the first element,
- 1 when placing the second element,
- 2 for the third element,
- 3 for the fourth element,
- 3 for the fifth attempt leading to a collision.

So we will use the first one as a representative.

The case of collision occurring when five elements were placed has three distinct cases:

1) 0, 1, 2, 3, 4 – giving 12 comparisons,
2) 0, 1, 2, 3, 5 – giving 11 comparisons,
3) 0, 1, 2, 3, 8 – giving 10 comparisons.

The case of collision occurring when six elements were placed has three distinct cases:

1) 0, 1, 2, 3, 4, 5 – giving 14 comparisons,
2) 0, 1, 2, 3, 4, 8 – giving 13 comparisons,
3) 0, 1, 2, 3, 5, 8 – giving 12 comparisons.

Similarly, we can determine the number of comparisons for the remaining cases, the results are summarized in the table II.

TABLE II
MINIMAL AND MAXIMAL NUMBER OF COMPARISONS WHEN FILLING THREE OF HEIGHT 4

no. of elem. in the tree	prob. of collision	min. number	max. number
4	2/15	9	9
5	1/5	10	12
6	13/63	12	14
7	5/28	14	17
8	295/2268	16	20
9	232/2835	19	21
10	41/945	22	23
11	1/54	25	26
12	1/162	28	29
13	2/1323	31	31
14	2/8505	34	34
15	1/59535	37	37

From data in the table II, we can calculate the expected number of comparisons per element:

- $\frac{42591761}{12039300} \approx 3.5$ in the minimal case,
- $\frac{11680088933}{2860537680} \approx 4.1$ in the maximal case.

So, we will continue with the value 4.

The first element, placed on the 0 node, is compared with $\frac{1555849}{238140} \approx 6.5$ elements, on average.

By comparison, we meant the above simple arithmetic comparison, which means that the most significant bits of both blocks are compared first. If they are equal, then the second most significant bits are compared, and so on.

The probability that the comparison reaches the t-th most significant bit and ends there is

$$p(t) = \begin{cases} 2^{-t}, & t = 1, ..., d-1, \\ 2^{1-d}, & t = d, \end{cases}$$

so after c comparisons, probability that at least one of them reaches t-th most significant bit and not further equals

$$p_c(t) = \begin{cases} (1-2^{-t})^c - \left(1-2^{1-t}\right)^c, & t = 1, ..., d-1, \\ 1 - \left(1-2^{1-d}\right)^c, & t = d. \end{cases}$$

From this formula, we can numerically calculate the expected number of bits at least once used in any comparison. For $d = 32$ and four comparisons, we obtained 3.5 bits. Even for the exaggerated case of the first element, assuming seven comparisons, we only get 4.2. This means that even after performing an additional three iterations of the test on a sequence of cyclically rotated words, more than half of the bits will not affect the result.

The easiest to implement workaround is to perform eight iterations for words rotated by 4 bits. This will allow us to

achieve almost exact testing coverage while maintaining the correlation of the results of subsequent iterations at a low level.

The seemingly obvious solution of operating on blocks of 4 bits reduces the expected value of the number of active bits to 3.0. What is worse, it leads to a large unevenness of collision probabilities for individual leaves – from 8.7 to 16.9%.

A block length of 8 should be used instead, with an additional rotation by 4 bits. It gives almost the same coverage as a 32-bit block with minimal differences between collision probabilities for leaves. Furthermore, Pearson's correlation coefficient for two bytes overlapping by 4 bits is only 0.062, so it would be reasonable to perform this test in a single iteration for a sequence of overlapping bytes. In such a setup, it would be possible to implement it even in FPGAs.

For those who were convinced, we present the necessary probabilities below.

For a collision occurring on a given leaf:

```
0.12243053, 0.12494422, 0.12406619,
0.12660593, 0.12339690, 0.12592336,
0.12504015, 0.12759272,
```

probability of collision occurring when $k = 0..15$ elements where already placed are:

```
0.0, 0.0, 0.0, 0.0, 0.13334351,
0.20000763, 0.20635446, 0.17857414,
0.13006809, 0.08182863, 0.04337917,
0.01851266, 0.00616964, 0.00151047,
0.00023484, 0.00001676.
```

V. Conclusion

As shown, the method of determining eigenvectors for all eigenvalues of the covariance matrix seems important for two reasons. First, it allows for avoiding repetitions of the sequence evaluation for consecutive vector lengths and a significant acceleration of calculations. An additional bonus, described in our earlier work [8] is the possibility of determining the test statistic values for all vector dimensions, from 1 to the assumed t, after one run of the sequence.

Presented methods for analytically determining probability distributions used in Tree Filling Tests implemented in the Dieharder package give formulas for discrete cases, not just continuous ones, and trees of any height can be derived. We also point out several errors made while implementing both tests.

The implementation of the presented results will not drastically improve the quality of the assessments obtained using these tests, but it will improve their reliability and, in the second case, increase the detection power of the test.

Further work should lead to strict algebraic proof of the correctness of the eigenvectors' construction.

References

[1] R. G. Brown, "Dieharder: A random number test suite." [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php

[2] G. Marsaglia, "The marsaglia random number cdrom including the diehard battery of tests of randomness," 1995. [Online]. Available: web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/

[3] I. Good, "The serial test for sampling numbers and other tests for randomness," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 49, pp. 276–284, 1953. [Online]. Available: https://doi.org/10.1017/S030500410002836X

[4] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *NIST Special Publication SP 800-22 Revision 1a*, 2010. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-22r1a

[5] A. Alhakim, J. Kawczak, and S. Molchanov, "On the class of nilpotent markov chains, i. the spectrum of covariance operator," *Markov Processes and Related Fields*, vol. 4, pp. 629–652, 01 2004.

[6] A. Alhakim, "On the eigenvalues and eigenvectors of an overlapping markov chain," *Probability Theory and Related Fields*, vol. 128, pp. 589–605, 04 2004. [Online]. Available: https://doi.org/10.1007/s00440-003-0321-z

[7] P. Billingsley, *Probability and Measure" (3rd ed.).* John Wiley & Sons, 1995. [Online]. Available: www.colorado.edu/amath/sites/default/files/attached-files/billingsley.pdf

[8] K. Mańk, "Test czestości dla nakładajacych sie wektorów (in polish: Frequency test for overlapping vectors)," *Cyberprzestepczość i ochrona informacji – Bezpieczeństwo w internecie tom II, Wydawnictwo Wyższej Szkoły Menedżerskiej w Warszawie*, 2013.

[9] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, pp. 1–40, 01 2007. [Online]. Available: https://doi.org/10.1145/1268776.1268777

[10] G. P. E. and M. S. Nikulin, *A guide to chi-squared testing*. Wiley, 1996.