INFORMATICA, 2013, Vol. 24, No. 3, 339–356
© 2013 *Vilnius University*

A New Randomness Test for Bit Sequences

Pedro María ALCOVER[1]*, Antonio GUILLAMÓN[2],
María del Carmen RUIZ[3]

[1]*Department of Information Technologies and Comunications*
Universidad Politécnica de Cartagena
University Centre of Defense at the Spanish Air Force Academy, MDE-UPCT, Spain
[2]*Department of Applied Mathematics and Statistics*
Universidad Politécnica de Cartagena
University Centre of Defense at the Spanish Air Force Academy, MDE-UPCT, Spain
[3]*Department of Applied Mathematics and Statistics,*
Universidad Politécnica de Cartagena, Spain
e-mail: pedro.alcover@upct.es, antonio.guillamon@upct.es, maricarmen.ruiz@upct.es

Received: July 2011; accepted: October 2012

Abstract. Generating sequences of random numbers or bits is a necessity in many situations (cryptography, modeling, simulations, etc...). Those sequences must be random in the sense that their behavior should be unpredictable. For example, the security of many cryptographic systems depends on the generation of unpredictable values to be used as keys. Since randomness is related to the unpredictable property, it can be described in probabilistic terms, studying the randomness of a sequence by means of a hypothesis test. A new statistical test for randomness of bit sequences is proposed in the paper. The created test is focused on determining the number of different fixed length patterns that appear along the binary sequence. When 'few' distinct patterns appear in the sequence, the hypothesis of randomness is rejected. On the contrary, when 'many' different patterns appear in the sequence, the hypothesis of randomness is accepted.

The proposed can be used as a complement of other statistical tests included in suites to study randomness. The exact distribution of the test statistic is derived and, therefore, it can be applied to short and long sequences of bits. Simulation results showed the efficiency of the test to detect deviation from randomness that other statistical tests are not able to detect. The test was also applied to binary sequences obtained from some pseudorandom number generators providing results in keeping with randomness. The proposed test distinguishes by fast computation when the critical values are previously calculated.

Key words: randomness tests, cryptography, entropy, random bit generators.

1. Introduction and Problem Statement

The necessity of testing the randomness of a bit sequence arises in many fields, such as modeling, simulations and cryptography.

For example, the security of many cryptographic systems depends on the generation of unpredictable values to be used as keys, passwords, challenges or unique identifiers,

*Corresponding author.

see Koeune (2005), Eastlake *et al.* (2005), among others. Hence, the use of appropriate random number (bit) generators becomes a necessity. While there are many techniques to generate random bit sequences, in the context of cryptography the quality of each generator should be measured by means of different statistical tests. "Passing statistical tests is a necessary condition, since any statistical defect can be used by an attacker to gain information about the future output", as it is stated in Koeune (2005).

The importance of providing randomness statistical tests is also pointed out in some recents reports, such as Killmann and Schindler (2011) by the Federal Office for Informatic Security (BSI, Germany) and CCN-STIC-807 (2011) by the National Intelligence Centre (CNI, Spain). The former states that "many security mechanisms need secrets, e.g., cryptographic keys or authentication data. Unpredictable random numbers are ideal secrets for IT security applications". The latter states that "in cryptography, it is imperative to subject the generated pseudorandom sequence to all possible statistical tests before accepting it for cryptographic applications".

Generators of random sequences can be classified in two groups: random number generators and pseudorandom number generators. The first type uses a non-deterministic source together with some processing function to produce randomness. Some examples of the entropy source are the elapsed time between keystrokes or mouse movement, and the noise in an electrical circuit. The second type uses one or various seeds and a deterministic function to generate the sequence. There are many papers dealing with the design of random bit generators, see Jakobsson *et al.* (1998), Nasir (2009), Patidar *et al.* (2009), Peris-Lopez *et al.* (2010), Schindler (2009), among others.

In both cases, randomness statistical tests are required. When a random number generator is used, it is necessary to test the randomness of the outputs in order to determine that the physical source is really random. On the other hand, when a pseudorandom number generator is used, it is necessary to test that the sequence is forward unpredictable. Also, it should be back unpredictable since each value of the sequence is reproducible from its seed. This topic is tackled in Maurer (1992), Menezes *et al.* (1996), Rukhin (2000), Ryabko (2004), Doganaksoy (2006), Colbeck and Kent (2011), Hernández *et al.* (2004), Alani (2010) and Hamano and Yamamoto (2010).

Various statistical test suites have been developed to study the randomness of a sequence, such as Diehard, Crypt-XS and NIST. The Diehard suite was developed by Professor George Marsaglia (1995), the Crypt-XS suite was developed by researchers at the Information Security Research Center at Queensland University of Technology in Australia (Crypt-XS, 1998) and the NIST suite is the result of collaborations between the Computer Security Division and the Statistical Engineering Division at NIST (2010). All of them have been used to evaluate random and pseudorandom number generators for cryptographic applications.

It is important to emphasize that it is impossible to prove whether a generator is indeed a random bit (number) generator. Each test determines whether a sample sequence verifies a probability property that a truly random sequence would have. For example, a random bit sequence should have approximately the same number of ones and zeroes. However, there are non-random bit sequences with the same number of ones and zeroes (a sequence of size n with the first $\frac{n}{2}$ values equal to one and the rest $\frac{n}{2}$ values equal to zero, with n even). Separately, each test included in a test suite (for example NIST) is not highly efficient, but all tests together are a useful tool (but not infallible) to study the randomness of a sequence. In other words, given a non-random sequence, some statistical tests included in the test suite can indicate randomness (so these test have failed) whereas the rest of tests indicate non-randomness. Therefore, each randomness statistical test has its strengths and weaknesses.

As we mentioned above, randomness is related to the unpredictable property and it can be described in probabilistic terms. Generally, the study of the randomness of a sequence is performed by means of a hypothesis test (the null hypothesis assumes that the sequence is random), and it can be summarized as follows. Under the assumption that the sequence is random, the exact or approximate distribution of the test statistic is determined. Decision in the hypothesis test is taken through the p-value, which is computed using the theoretical distribution of the test statistic. If the p-value is less than the significance level α, then the sequence is considered non-random. Common values of α in cryptography are about 0.01 (99% confidence level).

In this paper, a new statistical randomness test (that we call the Topological Binary Test, TBT) is proposed for bit sequences. It can be considered as a complement of other statistical tests to detect deviation from randomness, so it helps to detect some weakness the generator may have. Its main strengths are the knowledge of the exact distribution for the test statistic, fast computing and that it can be applied to short or long sequences of bits.

The rest of the paper is organized as follows. In Section 2, the new statistical test is defined and its exact distribution is given. In Section 3, the simulation results of the proposed test are compared to those tests included in NIST. Finally, in Section 4 some conclusions are given.

2. Topological Binary Test

This section describes the proposed test (TBT), gives the exact distribution of the test statistic and points out some similarities and differences with other statistical randomness tests included in the NIST suite.

The statistical test suite NIST (version sts-2.1.1, August 2010) consists of 15 tests, each of them evaluating a necessary condition for the randomness in probabilistic terms. For instance, the frequency (monobit) test determines whether the number of ones and zeroes in a sequence are about the same. The frequency test within a block determines whether the frequency of ones in a m-bit block is approximately $m/2$. The run test evaluates whether the number of runs of ones and zeroes are as expected for a random sequence. These are necessary but not sufficient conditions to test randomness of a bit sequence.

As an example, let us consider a bit sequence of length $n = 10 \times 2^{10} = 10\,240$ as follows:

$$\{0, 0, 1, 1, 0, 0, \dots, 0, 0, 1, 1\} \tag{1}$$

with 5120 ones, 5120 zeroes and 5120 runs. Hence, the bit sequence is clearly non-random but monobit, frequency within a block and run tests would lead to accept the null hypothesis of randomness.

A set of five tests included in NIST are related to searching for m-bit patterns. For instance, the aim of the non-overlapping and overlapping template matching tests is the number of occurrences of pre-defined target substrings, whereas the Maurer's universal test focusses on the number of bits between matching patterns. For these three tests, approximate distributions of the test statistics are used and long sequences of bits are required. On the other hand, serial and approximate entropy tests focus on the frequency of every overlapping m-bit pattern.

The focus of the TBT is the number of distinct non-overlapping m-bit patterns across the entire sequence, where m represents the non-overlapping block length or pattern size. The test performance can be summarized as follows:

- **Step 1:** For a given sequence of bits of length n, $\{x_t\}_{t=1}^{n}$, select the size of the non-overlapping blocks m.
- **Step 2:** Partition the sequence into $k = [\frac{n}{m}]$ non-overlapping blocks of length m and discard the rest of the bits.
- **Step 3:** Consider the 2^m possible patterns of length m and determine the number of different m-bit patterns that appear across all the k blocks.
- **Step 4:** Compute the p-value or the critical value of the test using the exact distribution of the test statistic.

Clearly, a random bit sequence should contain "many" different m-bit patterns, so bit sequences with "few" distinct m-bit patterns must be non-random. Using the exact distribution of the test statistic, we can compute the critical value for each selection of block length, m, and number of blocks, k. That is, the minimum number of different m-bit patterns that should appear along the bit sequence to accept the null hypothesis of randomness.

Next example illustrates the performance of the TBT test. Let us consider again the bit sequence given in (1)

$$\{0, 0, 1, 1, 0, 0, \ldots, 0, 0, 1, 1\}$$

- **Step 1:** Let us select $m = 10$ the non-overlapping block length.
- **Step 2**: The bit sequence is then partitioned into $k = [\frac{10\times 2^{10}}{10}] = 2^{10} = 1024$ non-overlapping blocks of length 10, as follows:

$$\begin{array}{ccc}
\text{Block 1} & \rightarrow & [0, 0, 1, 1, 0, 0, 1, 1, 0, 0], \\
\text{Block 2} & \rightarrow & [1, 1, 0, 0, 1, 1, 0, 0, 1, 1], \\
\text{Block 3} & \rightarrow & [0, 0, 1, 1, 0, 0, 1, 1, 0, 0], \\
\vdots & & \vdots \\
\text{Block 1024} & \rightarrow & [1, 1, 0, 0, 1, 1, 0, 0, 1, 1],
\end{array}$$

- **Step 3:** There are only 2 different 10-bit patterns across all the blocks,

$$\begin{array}{lcl} \text{Pattern type 1} & \rightarrow & [0,0,1,1,0,0,1,1,0,0], \\ \text{Pattern type 2} & \rightarrow & [1,1,0,0,1,1,0,0,1,1], \end{array}$$

- **Step 4:** The critical value or the p-value of the test can be computed using the exact distribution of the test statistic. In this case, the critical value of the test for $m = 10$ and $k = 2^{10}$ is $cv = 624$ (see Table 1). Then, we have that:

$$\begin{array}{ccc} \text{number of different patterns} & & \text{minimum number of different patterns,} \\ \text{found along the sequence} & < & \text{to accept randomness (critical value),} \\ \downarrow & & \downarrow \\ 2 & < & 624 \end{array}$$

  and therefore the TBT test strongly conclude that the binary sequence given in (1) is not random, whereas some other statistical tests would accept randomness (for example, the monobit, block-frecuency, run tests and CumulativeSums included in NIST). This fact shows the utility of the TBT test to complement other statistical tests included in the suites.

The next results show the theoretical grounding of the proposed test and how to compute the critical value needed in Step 4.

Result 1. Let $\{x_t\}_{t=1}^n$ be a binary sequence of length n, let m be the selected block length or pattern size and k the number of non-overlapping blocks of length m in the sequence. Let W be the number of different m-bit patterns found along the k non-overlapping blocks. Then, the discrete probability function of W under the assumption of randomness is given by:

$$P(W = j) = \frac{(2^m)!}{(2^m - j)!} \cdot S_2(k, j) \cdot \left(\frac{1}{2^m}\right)^k \tag{2}$$

for $j = 1, 2, \ldots, \min(k, 2^m)$, where $S_2(k, j)$ denotes the Stirling number of second type.

Proof. When randomness is assumed, each m-bit pattern has the same probability to appear along the sequence. For the fixed block length, m, the number of possible m-bit patterns is 2^m:

$$\begin{array}{lcccc} & \text{Position 1} & \text{Position 2} & \ldots & \text{Position } m \\ \text{Block of size } m\text{:} & \downarrow & \downarrow & & \downarrow \\ & \text{2 posibilities} & \text{2 posibilities} & & \text{2 posibilities.} \end{array}$$

Then, under the assumption of randomness, each m-bit pattern has probability $\frac{1}{2^m}$ to appear.

As the sequence has been partitioned into k non-overlapping blocks of length m:

$$\begin{array}{lccl} \text{Block 1} & [x_1, x_2, \ldots, x_m] & \rightarrow & \text{Probab to appear} = \frac{1}{2^m}, \\ \text{Block 2} & [x_{m+1}, x_{m+2}, \ldots, x_{2m}] & \rightarrow & \text{Probab to appear} = \frac{1}{2^m}, \\ & \vdots & & \\ \text{Block } k & [x_{n-m+1}, x_{n-m+2}, \ldots, x_n] & \rightarrow & \text{Probab to appear} = \frac{1}{2^m}, \end{array}$$

the probability of finding j different patterns across all the k blocks is given by:

$$\begin{aligned} P(W=j) &= \sum_{i=1}^{C}\left(\frac{1}{2^m}\right)\cdot\left(\frac{1}{2^m}\right)\cdots\left(\frac{1}{2^m}\right) \\ &= C\cdot\left(\frac{1}{2^m}\right)^k, \end{aligned}$$

where C denotes the number of ways to put exactly j different m-bit patterns into the k blocks.

On the one hand, there are $\binom{2^m}{j}$ ways to select j distinct patterns among the 2^m possible m-bit patterns. On the other hand, there are $j!\cdot S_2(k,j)$ ways to find the selected j different patterns across the k blocks.

Therefore, C is given by:

$$C=\binom{2^m}{j}\cdot j!\cdot S_2(k,j)=\frac{(2^m)!}{(2^m-j)!}\cdot S_2(k,j),$$

where $S_2(k,j)$ is the Stirling number of second type, see (Canovas and Guillamon, 2009) for more details.

Result 2. The exact distribution function of W under the assumption of randomness is given by:

$$\begin{aligned} F_W(x) &= P(W\leqslant x) \\ &= \begin{cases} 0, & \text{if } x<1, \\ \sum_{j=1}^{[x]}\left(\frac{(2^m)!}{(2^m-j)!}\cdot S_2(k,j)\cdot\left(\frac{1}{2^m}\right)^k\right), & \text{if } 1\leqslant x<\min(k,2^m), \\ 1, & \text{if } x\geqslant\min(k,2^m). \end{cases} \end{aligned} \tag{3}$$

Proof. The proof is straightforward using Result 1 and that the distribution function of a discrete random variable can be obtained by adding the discrete probability function.

The TBT determines whether the number of different m-bit patterns found along the k non-overlapping blocks is as high as expected for truly random sequences. Thus, when few different patterns appear, the test will reject the null hypothesis of randomness. The p-value of the test is computed by:

$$p\text{-value}=F_W(x_0)=P(W\leqslant x_0),$$

where x_0 is the number of different patterns found for the given bit sequence. If p-value $<\alpha$ (significance level), the test concludes that the sequence is non-random because the number of different m-bit patterns found is lower than the expected for a random sequence (the critical value).

Tables 1 and 2 show the evolution of the critical value (cv) for different choices of m and k, with significance level $\alpha=0.01$. When $m=16$ or greater, some terms in (3) are

Table 1
Critical values of TBT test for $k = 2^m$.

m	cv	2^m	Proportion
8	150	$2^8 = 256$	0.586
9	307	$2^9 = 512$	0.600
10	624	$2^{10} = 1\,024$	0.609
11	1262	$2^{11} = 2\,048$	0.616
12	2543	$2^{12} = 4\,096$	0.621
13	5113	$2^{13} = 8\,192$	0.624
14	10264	$2^{14} = 16\,384$	0.626
15	20582	$2^{15} = 32\,768$	0.628
16	41241	$2^{16} = 65\,536$	0.629

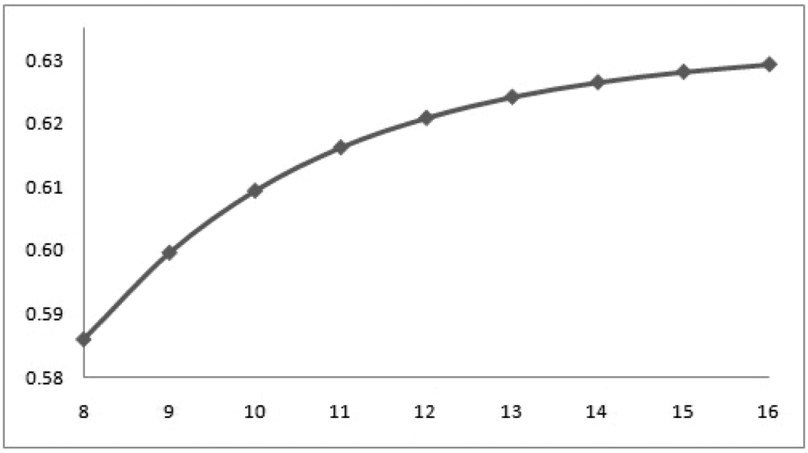

Fig. 1. Proportion of different m-bit pattern which corresponds to each critical value.

hard to compute (specially the Stirling numbers) and symbolic calculus is required. Hence, computation of the critical value for $m = 16$ was performed by a special computer with a high capacity cluster of 152 cores (Hipatia, 2011) under Mathematica 7.0. In Table 1, the number of non-overlapping m-blocks in the sequence (k) coincides with the number of possible m-bit patterns (2^m), whereas in Table 2 greater values of k are considered. If the number of different patterns found in the sequence is lower that the critical value, the TBT concludes that the sequence is non-random. Figure 1 shows (for each pattern size m) the proportion of different m-bit pattern which corresponds to each critical value. Note that this proportion increases when m get greater and we could guess an asymptotic behavior for $m \geqslant 16$ around the proportion 0.63. Therefore, a good approximation of the critical values for $m \geqslant 17$ could be $0.63 \cdot 2^m$. Figure 2 shows the proportions corresponding to each critical value for different number of non-overlapping blocks (k) proportional to 2^m. The graph reveals a similar behavior for different m values.

Table 2
Critical values of TBT test for greater k.

$m\backslash k$	$k = 2^m$	$k = 2 \cdot 2^m$	$k = 3 \cdot 2^m$	$k = 4 \cdot 2^m$	Possible m-patterns
$m = 8$	$cv = 150$ $p = 0.586$	$cv = 211$ $p = 0.828$	$cv = 239$ $p = 0.934$	$cv = 246$ $p = 0.961$	256
$m = 10$	$cv = 624$ $p = 0.609$	$cv = 864$ $p = 0.844$	$cv = 958$ $p = 0.936$	$cv = 995$ $p = 0.972$	1024
$m = 12$	$cv = 2543$ $p = 0.621$	$cv = 3499$ $p = 0.854$	$cv = 3862$ $p = 0.943$	$cv = 4001$ $p = 0.977$	4096

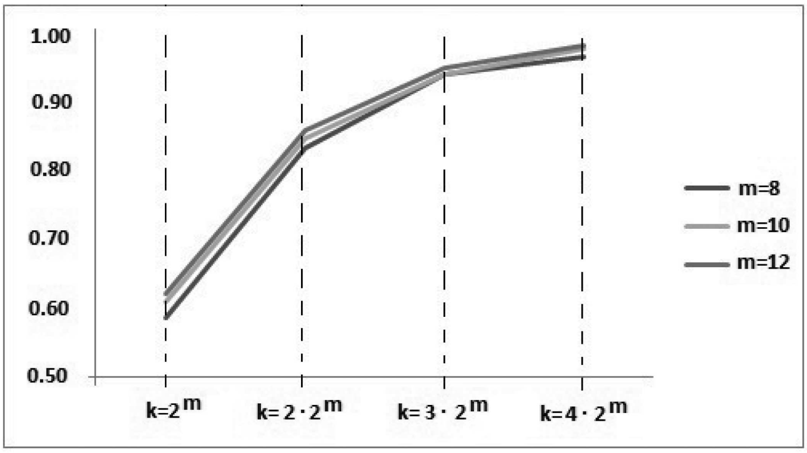

Fig. 2. Proportions corresponding to each critical value for non-overlapping blocks (k) proportional to 2^m.

3. Simulation Results

The proposed test (TBT) and those included in NIST were applied to 20 different pseudorandom number generators. Ten of the selected generators are recommended for simulation proposes because they have extremely long periods, low correlation and the strongest proof of randomness. They belong to the so-called second-generation *ranlux* generators and they are included in the Random Number Generator part of the GSL library (GSL Library, 2010). The rest of the selected generators belong to the set of *unix* random number generators (also implemented in GSL) and they are considered quite acceptable in spite of not producing high-quality randomness.

Fifty million values were generated with each pseudorandom number generator. Then, it was transformed into a sequence of bits by a parity criterion (the evenness or oddness of the number of bits with value one within a given set of bits), yielding 0 for even parity and 1 for odd parity. The same seed was used with all generators (the first nine decimal digits of $\sqrt{2}$) in order to provide reproducible sequences. Using a unique seed is not appropriate to test the efficiency of a pseudorandom number generator. However, this is not the aim of the paper, the real aim is to show that the performance of the proposed test is as good as other tests used for cryptographic applications.

Table 3
Block length (m) used in each statistical test (NIST and TBT) for different stream sizes (n).

	$n =$	$n =$	$n =$	$n =$	$n =$	$n =$	$n =$	$n =$
	491 520	229 376	106 496	49 152	22 528	10 240	4 608	2 048
BlockFreq	5000	2500	1100	500	250	110	50	21
Appr.Entrop	13	12	11	10	9	8	7	6
Serial test	16	15	14	13	12	11	10	9
TBT	15	14	13	12	11	10	9	8

Table 4
Block length (m) used in each statistical test (NIST and TBT) for length of the stream = 1 048 576.

Block length	BlockFreq	Non overlaping	Overlaping	Appr.Entrop
	10500	10	10	14
Block length	Serial test	Linear complexity	TBT	
	17	1000	16	

Different stream sizes and several number of streams were selected in order to compare results. Each bit stream size was of the form $n = m \cdot 2^m$ with $m = 8, 9, \ldots, 16$, where n represents the length of the stream. The number of selected streams were 47 samples (when $m = 16$), 100 (when $m = 15$), 200 (when $m = 14$), 400 (when $m = 13$), and 500 samples (when $m = 12, 11, 10, 9$, or 8). Therefore, the number of non-overlapping blocks of length m when using TBT coincides with the number of possible m-bit patterns ($k = 2^m$).

Recall that six of the statistical randomness tests included in NIST require a long sequence of bits to be applied (sequence length $\geqslant 10^6$ bits), whereas the rest of the tests (also TBT) can be used with shorter sequences. Thus, all tests were applied to the longest stream size and only some tests were applied to shorter stream sizes.

Tables 3 and 4 show the block length (m) and the bit stream size (n) selected for each statistical test according to NIST recommendations.

In all simulations the percentage of acceptance of the randomness hypothesis is computed, using the standard significance level $\alpha = 0.01$ (99% confidence level). Although the tests were applied to twenty different random number generators, the results of only three of them (*ranlux, mrg* and *rand48*) are included in the paper in order to simplify the writing.

Table 5 shows the results for the longest bit stream size, $n = 16 \cdot 2^{16} = 1\,048\,576$, with 47 samples. Remark that the TBT test provides a similar percentage of randomness acceptance than the other statistical tests.

Regarding computation time, notice that TBT is quite fast if the critical values have been previously calculated.

Tables 6, 7 and 8 contain the results for shorter bit stream sizes, where only some tests can be applied.

All tables reveal that the TBT test provides similar results than the other randomness statistical tests.

On the other hand, the statistical randomness tests were also applied to the sequences obtained as the binary expansion of the irrational numbers $\sqrt{2}$, e and π. For these cases,

Table 5
Results for a long bit stream size ($n = 1\,048\,576$).

	ranlux	mrg	rand48
Frequency	1.000	0.979	1.000
BlockFrequency	1.000	1.000	0.979
CumulativeSums (f)	1.000	0.979	1.000
CumulativeSums (b)	1.000	0.957	0.979
Runs	1.000	0.957	1.000
LongestRun	1.000	1.000	0.979
Rank	1.000	1.000	1.000
FFT	0.979	1.000	0.957
NonOverlappingTemplate	0.979	1.000	0.979
OverlappingTemplate	0.979	1.000	0.979
Universal	0.979	1.000	1.000
Appr.Entrop	0.979	0.915	0.957
RandomExcursions	0.941	0.968	1.000
RandomExcursionsVariant	1.000	1.000	1.000
Serial (∇^1)	1.000	0.957	1.000
Serial (∇^2)	1.000	1.000	1.000
LinearComplexity	1.000	0.957	0.957
TBT	**0.979**	**1.000**	**1.000**

Table 6
Results for *ranlux* using different bit stream sizes (n).

Statistical test	$n =$ 491 520	$n =$ 229 376	$n =$ 106 496	$n =$ 49 152	$n =$ 22 528	$n =$ 10 240	$n =$ 4 608	$n =$ 2 048
Frequency	0.990	0.980	0.988	0.992	0.996	0.998	0.992	0.996
BlockFreq	1.000	0.995	0.990	0.984	0.984	0.998	0.996	0.992
CumSums (f)	0.980	0.980	0.993	0.994	0.996	1.000	0.994	0.992
CumSums (b)	0.990	0.985	0.990	0.996	0.996	1.000	0.994	0.996
Runs	1.000	1.000	0.998	0.994	1.000	0.994	0.998	0.996
LongestRun	0.990	0.985	0.993	0.990	0.998	0.992	0.986	0.994
Rank	1.000	0.990	0.993	0.986	0.996	0.982	0.984	0.970
FFT	1.000	0.990	0.968	0.986	0.986	0.990	0.988	0.984
Appr.Entrop	0.990	0.980	0.988	0.982	0.990	0.976	0.992	0.990
Serial (∇^1)	0.990	0.990	0.998	0.988	0.998	0.988	0.994	0.996
Serial (∇^2)	1.000	0.990	0.993	0.984	0.994	0.986	0.996	0.994
TBT	**1.000**	**1.000**	**0.983**	**0.992**	**0.998**	**0.990**	**0.990**	**0.996**

the sequences of 10^6 bits included in NIST package were considered and stream sizes of $n = 8 \cdot 2^8$, $n = 9 \cdot 2^9$ and $n = 10 \cdot 2^{10}$ were selected with 400, 200 and 90 samples respectively.

NIST also contains a data file corresponding to a random bit sequence generated by Blum Blum Shub (BBS) pseudorandom number generator. The length of this bit sequence allows to apply all the statistical tests with parameters given in Tables 3 and 4.

Tables 9, 10, 11, 12 and 13 summarize the percentage of randomness acceptance for each statistical test and each stream size. Again, the TBT test provides similar results to the other statistical tests.

Table 7
Results for *mrg* using different bit stream sizes (n).

Statistical test	$n =$ 491 520	$n =$ 22 9376	$n =$ 106 496	$n =$ 49 152	$n =$ 22 528	$n =$ 10 240	$n =$ 4 608	$n =$ 2 048
Frequency	0.990	0.995	0.980	0.992	1.000	0.998	0.982	0.990
BlockFreq	0.990	0.990	0.995	0.994	0.994	0.994	0.996	0.992
CumSums (f)	0.990	0.995	0.988	0.992	0.998	0.998	0.986	0.984
CumSums (b)	0.990	0.985	0.990	0.996	1.000	0.998	0.982	0.990
Runs	0.970	0.985	0.993	0.984	0.996	1.000	0.988	0.982
LongestRun	0.990	0.995	1.000	0.990	1.000	0.994	0.990	0.988
Rank	1.000	1.000	0.995	0.994	0.996	0.998	0.980	0.982
FFT	1.000	0.995	0.995	0.984	0.980	0.986	0.998	0.988
Appr.Entrop	0.970	0.965	0.973	0.978	0.994	0.996	0.996	0.980
Serial (∇^1)	0.990	0.970	0.988	0.986	0.994	0.998	0.990	0.984
Serial (∇^2)	0.990	0.985	0.985	0.988	0.994	0.998	0.988	0.984
TBT	**0.980**	**1.000**	**0.998**	**0.996**	**0.992**	**0.996**	**0.990**	**0.992**

Table 8
Results for *rand48* using different bit stream sizes (n).

Statistical test	$n =$ 491 520	$n =$ 229 376	$n =$ 106 496	$n =$ 49 152	$n =$ 22 528	$n =$ 10 240	$n =$ 4 608	$n =$ 2 048
Frequency	1.000	0.990	0.995	0.992	0.996	0.992	0.990	0.996
BlockFreq	0.980	0.995	0.985	0.990	0.994	0.988	0.994	0.990
CumSums (f)	1.000	0.985	0.993	0.992	0.996	0.992	0.994	0.988
CumSums (b)	1.000	0.990	0.993	0.992	0.992	0.994	0.988	0.992
Runs	0.990	0.995	0.985	0.984	0.992	0.984	0.988	0.986
LongestRun	1.000	0.985	0.990	0.992	0.990	0.990	0.990	0.988
Rank	1.000	0.985	0.988	0.990	0.990	0.984	0.986	0.986
FFT	0.980	0.995	0.983	0.986	0.988	0.992	0.992	0.994
Appr.Entrop	0.970	0.965	0.978	0.984	0.972	0.990	0.990	0.980
Serial (∇^1)	1.000	0.980	0.993	0.992	0.994	0.990	0.988	0.990
Serial (∇^2)	1.000	0.990	0.990	0.988	0.990	0.982	0.980	0.988
TBT	**0.990**	**1.000**	**0.985**	**0.980**	**0.996**	**0.992**	**0.988**	**0.992**

Table 9
Results for the binary expansion of $\sqrt{2}$.

Statistical test	$n = 2\,048$ Smp = 400	$n = 4\,608$ Smp = 200	$n = 10\,240$ Smp = 90
Frequency	0.985	0.990	0.989
BlockFrequency	0.987	0.995	0.989
CumulativeSums (forward)	0.990	0.990	0.989
CumulativeSums (backward)	0.987	0.990	1.000
Runs	0.9875	0.990	1.000
LongestRun	0.9825	0.995	0.989
Rank	0.970	0.990	0.989
FFT	0.982	0.990	1.000
ApproximateEntropy	0.985	0.990	0.989
Serial (∇^1)	0.985	0.995	1.000
Serial (∇^2)	0.990	1.000	0.989
TBT	**0.990**	**0.995**	**0.989**

Table 10
Results for the binary expansion of e.

Statistical test	$n = 2\,048$ Smp = 400	$n = 4\,608$ Smp = 200	$n = 10\,240$ Smp = 90
Frequency	0.995	0.995	0.989
BlockFrequency	0.985	0.970	1.000
CumulativeSums (forward)	0.995	1.000	0.989
CumulativeSums (backward)	0.997	0.995	0.989
Runs	0.995	1.000	1.000
LongestRun	0.985	0.975	0.989
Rank	0.990	0.980	1.000
FFT	0.997	0.995	0.967
ApproximateEntropy	0.990	0.995	0.989
Serial (∇^1)	0.992	0.990	0.989
Serial (∇^2)	0.992	0.985	0.989
TBT	**0.997**	**0.995**	**1.000**

Table 11
Results for the binary expansion of π.

Statistical test	$n = 2\,048$ Smp =400	$n = 4\,608$ Smp = 200	$n = 10\,240$ Smp = 90
Frequency	0.992	0.995	1.000
BlockFrequency	0.990	0.995	0.989
CumulativeSums (forward)	0.985	0.990	0.989
CumulativeSums (backward)	0.990	0.995	1.000
Runs	0.997	1.000	0.978
LongestRun	0.990	0.975	1.000
Rank	0.982	0.980	0.978
FFT	0.995	0.980	0.956
ApproximateEntropy	0.982	0.995	1.000
Serial (∇^1)	0.985	0.990	1.000
Serial (∇^2)	0.992	0.995	1.000
TBT	**0.992**	**0.985**	**0.989**

Finally, the efficiency of the TBT test was analyzed for non-random sequences containing short period cycles.

On the one hand, let us consider a bit sequence consisting of 10 240 random bits (generated by *taus*) and repeated 100 times ($L = 10\,240 = 10 \cdot 2^{10}$ represents the period length). Certainly, all statistical tests will fail if the bit stream size is $n = 10 \cdot 2^{10}$, but we may wonder about what will happen for greater stream lengths.

Concerning the TBT test, we point out two possibilities to detect non-randomness: increasing m (pattern size) and n (stream size) simultaneously or just increasing n for a fixed m. In previous results, only bit stream sizes of the form $n = m \cdot 2^m$ where used, and for the TBT test, the number of non-overlapping blocks were given by $k = 2^m$. However, other values of k are possible and could be useful to detect short period cycles.

Table 14 shows the percentage of the randomness acceptance for different values of the bit stream size (n) and the bit pattern size (m). Results reveal the importance of those

Table 12
Results for a long bit stream size ($n = 1048\,576$) for BBS data.

Statistical test	$n = 1\,048\,576$
Frequency	0.979
BlockFrequency	1.000
CumulativeSums (f)	0.979
CumulativeSums (b)	0.979
Runs	0.979
LongestRun	1.000
Rank	0.979
FFT	0.957
NonOverlappingTemplate	1.000
OverlappingTemplate	0.957
Universal	1.000
Appr.Entrop	0.957
RandomExcursions	1.000
RandomExcursionsVariant	1.000
Serial (∇^1)	1.000
Serial (∇^2)	1.000
LinearComplexity	1.000
TBT	**1.000**

Table 13
Results for *BBS* using different bit stream sizes (n).

Statistical test	$n =$ 491 520	$n =$ 229 376	$n =$ 106496	$n =$ 49 152	$n =$ 22528	$n =$ 10240	$n =$ 4608	$n =$ 2048
Frequency	0.990	0.995	0.988	0.994	0.986	0.988	0.982	0.986
BlockFreq	0.990	1.000	0.988	0.994	0.986	0.994	0.996	0.992
CumSums (f)	0.990	0.995	0.988	0.996	0.990	0.988	0.980	0.982
CumSums (b)	0.980	0.995	0.990	0.994	0.990	0.988	0.974	0.984
Runs	0.990	1.000	0.995	0.980	0.990	0.992	0.992	0.996
LongestRun	1.000	1.000	0.998	0.986	0.998	0.992	0.994	0.988
Rank	0.980	1.000	0.998	0.982	0.998	0.998	0.986	0.982
FFT	0.990	0.980	0.988	0.984	0.982	0.984	0.986	0.974
Appr.Entrop	0.990	0.995	0.978	0.974	0.978	0.974	0.986	0.988
Serial (∇^1)	1.000	0.995	0.988	0.992	0.990	0.986	0.992	0.994
Serial (∇^2)	0.990	1.000	0.990	0.992	0.992	0.988	0.984	0.992
TBT	**1.000**	**1.000**	**0.995**	**0.998**	**0.996**	**0.992**	**0.990**	**0.992**

measures in determining non-randomness. Note that the TBT test requires $m = 12$ and $n = 49\,152$ to detect the cycles. Nevertheless, six of the statistical tests still conclude randomness even for $n = 49\,152$.

The cycles can also be detected by the TBT test increasing the bit stream size n and keeping m fixed. For example, if $m = 10$ and $n = 20\,480$, then 652 different 10-bit patterns were found in the sequence whose corresponding critical value is 864 (see Table 2). Therefore, the TBT concludes that the sequence is non-random for $m = 10$ and $n = 20\,480$. Likewise, the same conclusion is obtained for $m = 10$ and greater n.

On the other hand, let us consider some bit sequences obtained using a congruential generator with no-maximal period. The sequences were generated by the following

Table 14
Results for a non-random bit sequence with many cycles.

Statistical test	$n = 10\,240$ Smp = 100	$n = 22\,528$ Smp = 45	$n = 49\,152$ Smp = 20
Frequency	1.000	1.000	1.000
BlockFrequency	1.000	1.000	1.000
CumulativeSums (forward)	1.000	1.000	1.000
CumulativeSums (backward)	1.000	1.000	1.000
Runs	1.000	1.000	1.000
LongestRun	1.000	0.000	0.000
Rank	1.000	1.000	1.000
FFT	1.000	0.000	0.000
ApproximateEntropy	1.000	0.000	0.000
Serial (∇^1)	1.000	0.000	0.000
Serial (∇^2)	1.000	0.000	0.000
TBT	**1.000**	**1.000**	**0.000**

Table 15
Results for non-random sequences with a part of a cycle.

Statistical test	$n = 106\,496$ $a = 53$, $c = 6$ Period_Cycle = 53 248	$n = 49\,152$ $a = 25$, $c = 10$ Period_Cycle = 24 576
Frequency	Random	Random
BlockFrequency	Random	Random
CumulativeSums (forward)	Random	Random
CumulativeSums (backward)	Random	Random
Runs	Random	**Non-Random**
LongestRun	Random	**Non-Random**
Rank	Random	Random
FFT	**Non-Random**	**Non-Random**
ApproximateEntropy	**Non-Random**	**Non-Random**
Serial (∇^1)	**Non-Random**	**Non-Random**
Serial (∇^2)	**Non-Random**	**Non-Random**
TBT	**Non-Random**	**Non-Random**

equation:

$$x_{n+1} = (a \cdot x_n + c) \bmod L$$

using different values of the parameters. Note that the selection of appropriate parameters provides random bit sequences (maximal period). However, for $L = 106\,496$, the selection of $a = 53$ and $c = 6$ provides a cycle of period 53 248, and for $L = 49\,152$, the selection of $a = 25$ and $c = 10$ provides a cycle of period 24 576. The results of each test for the two non-random bit sequences are given in Table 15. These two bit sequences are of size 106 496 and 49 152 respectively, so that only one sample was tested in each case.

Note that the TBT test concludes that both sequences are non-random whereas some other statistical tests fail.

4. Conclusions

It is well known that random and pseudorandom number generators are incorporated in many IT products and play an important role in numerous cryptographic applications. For example, the security of many cryptographic systems depends on the generation of unpredictable values. The quantities generated must be of sufficient size and be random in the sense that the probability of any particular value being selected must be sufficiently small to avoid that an adversary could gain advantage through optimizing a search strategy based on such probability.

It is impossible to give a mathematical proof that a generator is indeed a random bit generator. Each test determines whether a sample sequence verifies a probability property that a truly random sequence would have, so they help detect certain kinds of weaknesses the generator may have. Passing the statistical tests is a necessary but not sufficient condition for a sequence to be random. In other words, the quality of each generator should be measured by means of different statistical tests, since any statistical defect can be used by an attacker to gain information about the future output. Therefore, it is necessary to subject the generated sequence to all possible statistical tests before accepting it for cryptographic applications.

Some examples of statistical test suites that have been developed to study the randomness of a sequence are Diehard, Crypt-XS and NIST, among others. They include a wide range of statistical tests that have been used to evaluate random and pseudorandom number generators for cryptographic applications.

In this paper, a new statistical test (called TBT) for randomness of bit sequences is proposed. The TBT test can be used as a complement of other statistical tests included in suites to study randomness, as we mentioned above.

The test developed in this paper is based on determining the number of different m-bit patterns that appear in the sequence, for a fixed value m. When 'few' distinct m-bit patterns are found along the sequence the hypothesis of randomness is rejected, whereas 'many' different patterns appearing in the sequence lead to the acceptance of randomness.

Basic notions on probability allow to get the exact distribution of the test statistic and to determine the minimum number of different m-bit patterns that should appear along the bit sequence to accept the null hypothesis of randomness. The proposed test provided results in agreement with randomness when it was applied to binary sequences obtained from some standard pseudorandom number generators. On the other hand, simulation results showed the efficiency of the test to detect deviation from randomness that other statistical tests (implemented in some widely used suites) are not able to detect.

The main strengths of the created test are that it can be applied to short and long sequences of bits and its fast computation when the critical value are previously calculated.

Acknowledgements. We would like to thank the referees for their valuable comments which have been very useful to improve the paper.

The first author have been partially supported by the Spanish CICYT project EXPLORE (TIN2009-08572). The second author have been supported by the grant

ENE2010-20495-C02-02 and MTM2011-23221 [Ministerio de Ciencia e Innovación] and third author has been partially supported by the grant ENE2010-20495-C02-02 [Ministerio de Ciencia e Innovación].

References

Alani, M.M. (2010). Testing randomness in ciphertext of block-ciphers using DieHard tests. *International Journal of Computer Science and Network Security (IJCSNS)*, 10(4), 53–57.

Canovas, J.S., Guillamon, A. (2009). Permutations and time series analisys. *Chaos*, 19, 043103.

CCN-STIC-807 (2011). *Criptografía de empleo en el Esquema Nacional de Seguridad. Esquema Nacional de Seguridad*. Ministerio de Defensa. Spanish Goverment. `https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/800-Esquema_Nacional_de_Seguridad/807/_807-Criptologia_de_empleo_ENS-sep11.pdf`.

Colbeck, R., Kent, A. (2011). Private randomness expansion with untrusted devices. *Journal of Physics A: Mathematical and Theoretical*, 44(9), 095305. DOI:10.1088/1751-8113/44/9/095305.

Doganaksoy, A., Çalik, Ç., Sulak, F., Turan, M.S. (2006). New randomness tests using random walk. In: *National Cryptology Symposium II*, Ankara.

Eastlake, D., Schiller, J., Crocker, S. (2005). Randomness requirements for security. BCP 106, RFC 4086. `http://tools.ietf.org/html/rfc4086#section-1`.

GSL Library (2010). *Reference Manual, Edition 1.14, for GSL Version 1.14*, 4 March 2010. `http://www.gnu.org/software/gsl/manual/html_node/Random-Number-Generation.html`.

Hamano, K., Yamamoto, H. (2010). A randomness test based on T-complexity. *IEICE Transactions on Fundamentals*, 93(A), 7, 1346–1354.

Hernández, J.C., Sierra, J.M., Seznec, A. (2004). The SAC test: a new randomness test, with some applications to PRNG analysis. In: *Proceedings of the International Conference Computational Science and Its Applications (ICCSA)*, pp. 960–967.

Hipatia. Servicio de Apoyo a la Investigación Tecnológica: Universidad Politécnica de Cartagena. `http://www.upct.es/sait/sedic/servcalc.html`.

Information Security Institute (1998). Crypt-XS. `http://www.isi.qut.edu.au/resources/cryptx/`.

Jakobsson, M., Shriver, E., Hillyer, B.K., Juels, A. (1998). A practical secure physical random bit generator. In: *Proceedings of the 5th ACM Conference on Computer and Communications Security*. ACM, New York. DOI:10.1145/288090.288114.

Killmann, W., Schindler, W. (2011). A proposal for: functionality classes for random number generators. Version 2.0, 18 September 2011. `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/ AIS31_Functionality_classes_for_random_number_generators.pdf`.

Koeune, F. (2005), *Encyclopedia of Cryptography and Security*. In: H. van Tilborg (ed.), *Pseudo-Random Number Generator*. Springer, Berlin, pp. 485–487.

Marsaglia, G. (1995). *Diehard Battery of Tests of Randomness*. `http://stat.fsu.edu/~geo/diehard.html`.

Maurer, U. (1992). A universal statistical test for random bit generators. *Journal of Cryptology*, 5(2), 89–105.

Menezes, A.J., van Oorschot, P.C, Vanstone, S.A. (1996). *Handbook of Applied Cryptography*, CRC Press.

Nasir, Q. (2009). True random bit generator using ZCDPLL based on TMS320C6416. *International Journal of Communications, Network and System Sciences*, 4, 249–324.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. Special Publication SP 800-22, Revision 1a. `http://www.nist.gov`.

Patidar, V., Sud, K.K., Pareek, N. K. (2009). A pseudo random bit generator based on chaotic logistic map and its statistical testing. *Informatica*, 33, 441–452.

Peris-Lopez, P., San Millán, E., Van der Lubbe, J.C.A., Entrena, L.A. (2010). Cryptographically secure pseudo-random bit generator for RFID tags. In: *International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 1–6.

Ryabko, B.Ya., Stognienko, V.S., Shokin, Yu.I. (2004). A new test for randomness and its application to some cryptographic problems. *Journal of Statistical Planning and Inference*, 123, 365–376.

Rukhin, A.L. (2000). Approximate entropy for testing randomness. *Journal of Applied Probability*, 37, 88–100.

Schindler, W. (2009). Random number generators for cryptographic applications. In: *Cryptographic Engineering*, Springer, Berlin, pp. 5–23.

P. Alcover received his Bachelor's degree in Physics education at Universidad de Valencia in 1987, and his Doctorate degree in Computer Systems at Universidad de Murcia in 2003. At present he is Associate Professor in the Department of Information Technologies and Comunications at Universidad Politécnica de Cartagena. His main research interests include Cryptography, MDE (Model Driven Engineering), behavior modeling, models and models transformation, ATL (Atlas Transformation Language), JET, robotics, software engineering, computer vision.

A. Guillamón received his Bachelor's degree in mathematics education in 1990, and his Doctorate degree in Mathematics at Universidad de Murcia in 1997. At present he is Associate Professor in the Department of Applied Mathematics and Statistics at Universidad Politécnica de Cartagena. His main research interests include the nonparametric estimation, time series analysis, survival and reliability analysis and chaos.

M.C. Ruiz received her Bachelor's and Doctorate degrees in mathematics at Universidad de Murcia in 1998 and 2002 respectively. At present she is Associate Professor in the Department of Applied Mathematics and Statistics at Universidad Politécnica de Cartagena. Her main research interests include time series analysis, survival and reliability analysis and order statistics.

Naujas binarinių sekų atsitiktinumo testas

Pedro María ALCOVER, Antonio GUILLAMÓN, María del Carmen RUIZ

Atsitiktinių sekų generavimas yra reikalingas daugelyje sričių (kriptografijoje, modeliavime ir pan.). Tos sekos turi būti atsitiktinės ta prasme, kad jų elgesys turėtų būti nenuspėjamas. Pavyzdžiui, daugelio kriptografinių sistemų saugumas priklauso nuo nenuspėjamų reikšmių, naudojamų kriptografiniams raktams, generavimo. Kadangi atsitiktinumas yra susijęs su nenuspėjamumo savybe, jis gali būti aprašytas tikimybinėmis sąvokomis, tiriant sekų atsitiktinumą statistiniais testais. Straipsnyje pasiūlytas naujas atsitiktinumo testas. Sudarytas testas naudojasi fiksuoto ilgio šablonais, kurie pasitaiko binarinėje sekoje. Jei tik keli skirtingi šablonai pasitaiko sekoje, atsitiktinumo hipotezė atmetama. Kitaip, jei sekoje yra daug skirtingų šablonų, atsitiktinumo hipotezė yra priimama. Šis testas gali būti panaudojamas kaip papildomas standartiniams statistiniams testams. Tikslus statistikos skirstinys yra išvestas, ir todėl jis gali būti taikomas trumpoms bei ilgoms sekoms. Modeliavimo rezultatai parodė, kad testo efektyvumas nustatant atsitiktinumą yra aukštesnis negu kitų testų. Testas buvo tiriamas testuojant pseudoatsitiktines sekas. Be to, šis testas yra labai greitai skaičiuojamas, jei kritinės reikšmės yra suskaičiuotos iš anksto.