A remark on the NIST 800-22 Binary Matrix Rank Test

Nicu NECULACHE * Vlad-Andrei PETCU † Emil SIMION ‡

January 2022

Abstract

Statistical testing is a mechanism that has been included in various domains or fields, providing a method for making quantitative decisions about a particular sample. The statistical testing plays a big role in selecting and testing random and pseudorandom generators whose output may be used in the field of cryptography, specifically for the encryption, decryption and the keys or sub-keys generation. In this paper we study one of the NIST 800-22 [1] random number generation tests. We give an overview for the statistical testing and its importance for cryptography, then we focus on one of the tests, specifically the Binary Matrix Rank Test. We provide a logical schema and a new code implementation in *Python 3*. Further we evaluate the test, by running it on a collection of well chosen test samples and gathering the results based on which we do an assumption. More exactly, we validate if the binary sequence input can be classified as random or not depending on the bits density.

Keywords: statistical testing, random bit generator, bits density, binary matrix rank, P-value

1 Introduction

Statistical testing is a mathematical technique for analysing an algorithm or system based on some input-output pairs, usually the inputs being grouped into samples. The logical flow consists of running the algorithm (or the system) multiple times, on a significant inputs collection, obtaining the results and analysing them in order to classify/validate the algorithm. More specifically, the outcome is the acceptance or the rejection of the hypothesis that is made about the observed samples. Due to its efficiency on establishing the ownership of a set of independent observations, or measurements, to a specific population or probability distribution, statistical testing was adopted by many domains like healthcare, culture, economics, banking, cryptography and many more.

Mathematically speaking, a statistic test describes how closely the distribution of the data matches the distribution predicted under the null hypothesis of the statistical test used. The distribution of data is how often each observation occurs, and can be described by its central tendency and variation around that central tendency.

*Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email: nicu.neculache@gmail.com
†Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Email: petcuvlad92@gmail.com
‡Politehnica University of Bucharest, Email: emil.simion@upb.ro

As mentioned above, the statistical tests are commonly used in the field of cryptography, which is of interest to this paper - specifically for the encryption, decryption and the keys or sub-keys generation. Usually, these three processes are strongly dependent on the randomness of the used algorithms (or on the strength of sequence generated against cryptography analysis). This aspect brings up the importance for estimating the entropy, which is a measure of the amount of information needed for an attacker to find the encryption key or to predict the nonce values. If one statistical test finds some predictable information in the analyzed sample, then it will reject the null hypothesis.

The Statistical Test Suite developed by NIST is an excellent and exhaustive document looking at various aspects of randomness in a *long sequence of bits*. There are documented 15 statistical tests and in each test it adopted first a procedure to find the statistic of the *chi-square* variation χ^2 of a particular parameter for a given bits sequence with that obtained from the theoretical studies of an identical sequence under the assumption of randomness. It then adopted a technique to transform the χ^2 data to a randomness probability data, named as $P - value$.
Among these tests is found the Binary Matrix Rank Test which is is of interest and documented in this paper. Our work provides the mathematical background of the test in *Section 2.1* together with the new implementation and the logical schema in *Section 2.2*. Further, in *Section 3* we show our experiments results, based on which we correlate the acceptance of the hypothesis with the density of the bits in the given bits sequence.

2 Binary Matrix Rank Test

The focus of the Binary Matrix Rank Test, as the name suggests, is the rank of disjoint binary sub-matrices of the given bits sequence. The purpose of this test is to check for linear dependence among fixed length sub-strings of the original sequence. The main idea is to construct matrices of successive zeroes and ones from the sequence, and check for linear dependence among the rows or columns of the constructed matrices. The deviation of the rank - or rank deficiency - of the matrices in respect to the theoretically expected value gives the statistic of interest.

2.1 Mathematical fundamentals

For the mathematical fundamentals described in this section, we restructured the Section 3.5 from NIST SP 800-22 [1].

The bits sequence is restructured into N matrices of M rows and Q columns, for which the rank is calculated. The result states that the rank R of a $M \times Q$ random binary matrix takes values $r = 0, 1, 2, \ldots, m$, where $m \equiv min(M, Q)$, with probabilities

$$p_r = 2^{r(Q+M-r)-MQ} \prod_{i=0}^{r-1} \frac{(1-2^{i-Q})(1-2^{i-M})}{1-2^{i-r}}.$$

The probability values are fixed in the test suite code, as in our implementation, for $M = Q = 32$. The number M is a parameter of this test, given the fact that N is the new "sample size" and n is the length of the bits sequence, so ideally $n = M^2N$. In practice, values for M and N are chosen so that the

discarded part of the string, $n - NM^2$, is very small.

In this case, the rational is that $p_M \approx 0.2888...$, $p_{M-1} \approx 0.5776...$, $p_{M-2} \approx 0.1284...$ and all other probabilities are very small (≤ 0.005) when $M \geq 10$.

After obtaining the N square matrices, the binary rank R_l, $l = 1, ..., N$ is calculated and evaluated for each one of them, in order to determine the frequencies F_M, F_{M-1} and $N - F_M - F_{M-1}$:

$$F_M = \#\{R_l = M\}, \quad F_{M-1} = \#\{R_l = M - 1\}.$$

The reference distribution for the test statistic is a χ^2 distribution:

$$\chi^2 = \frac{(F_M - p_M N)^2}{p_M N} + \frac{(F_{M-1} - p_{M-1} N)^2}{p_{M-1} N} + \frac{(N - F_M - F_{M-1} - p_{M-2} N)^2}{p_{M-2} N}$$

The $P - value$ is $e^{-\chi^2(obs)/2}$. To be able to interpret the test, large values of $\chi^2(obs)$ indicate that the deviation of the rank distribution from that corresponding to a random sequence is significant. If the computed $P - value$ is < 0.01, then we can conclude that the sequence is non-random, otherwise we can assume that the sequence is random.

2.2 Implementation

In this section we present the logical schema, as shown in Figure 1 and the test implementation, together with our observations.

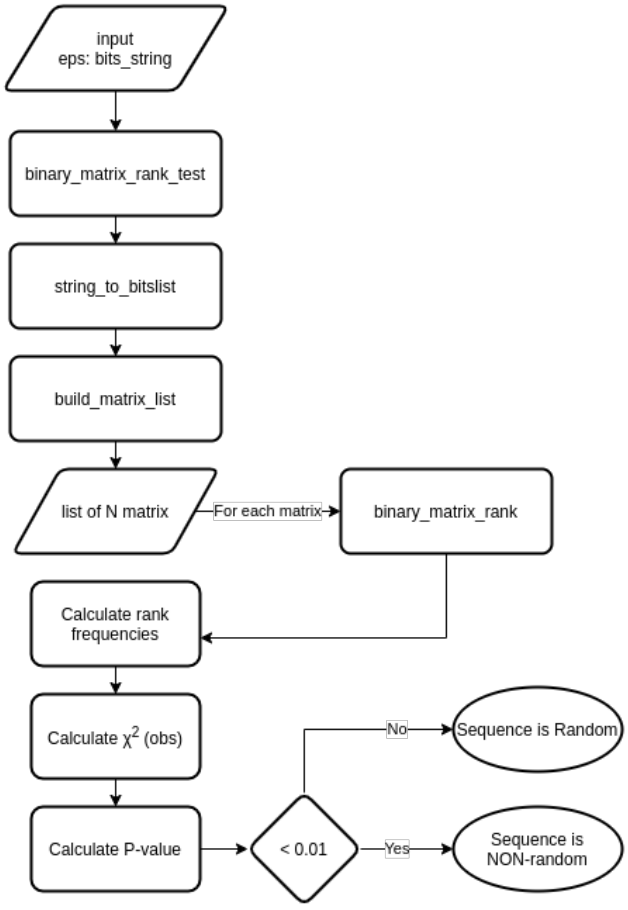

Figure 1: Logical schema

Following the above schema, the test algorithm takes as input a string of bits ϵ of length n and outputs the decision if the sequence is random or not. The algorithm uses the fixed values for $M = Q = 32$ and their corresponding probabilities.

The first process converts the string of bits into a numeric (boolean) list of values, this being the preparation for creating and filling up the actual matrices. The next process does the actual building of every $M \times Q$ matrix, where the bits are inlined as usual from left to right line by line. The number of resulted matrices is $N = \lfloor \frac{n}{MQ} \rfloor$ and the left bits are discarded.

The third process iterates every matrix and computes the binary rank R_l for each one, where $l = 1, ..., N$. The algorithm of computing the rank for a binary matrix are described in [1] Appendix F.1. Further, the matrices with the rank M, $M-1$ are counted to obtain the frequencies F_M, F_{M-1} and $N - F_M - F_{M-1}$. The algorithm uses $M = Q = 32$, so it uses the probabilities $p_M = 0.2888$, $p_{M-1} = 0.5776$, $p_{M-2} = 0.1284$ as mentioned in the previous section, based on which the $\chi^2(obs)$ and the $P - value$ are calculated.

The last process implies the comparison of the $P - value$ with the threshold (0.01) so the assumption of randomness can be made.

Our solution is implemented in *Python* and can be found on *Github* at [6].

3 Experimental results

In this section we study the variations of the test outputs, like the variation of $P - value$ in respect to the density of bits in the given sequence. This evaluation was made according to the choice of the input samples.

For the choice of the first sample, we used the random library from *Python* to generate bit by bit a set of strings, each one of length $n = 100000$. Then we analyzed how the $P - value$ varies in relation to the density of the bits.

The test processed 2100 random generated inputs, from which only 18 were classified as *non-random* (around $0,857\%$). As shown in Figure 2 we observed that the density of the bits varies between $49 - 51\%$ of the generated string length.

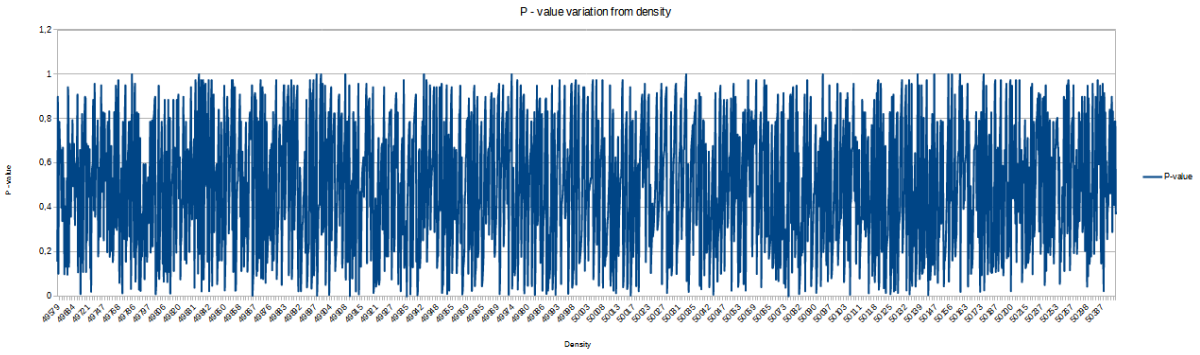

Figure 2: $P - value$ variation in respect to bits density (Python random library)

After the prior observations we made regarding the density percentage of the bits, using the first sample test, we chose to iterate by one the density over a range and generate customized random sequences for every iteration. As shown in Figure 3, the generated inputs of length 50000 have a variation of the bits density between 0 and 50000. Note that, the density in the figure can refer either to the density of the

bit 1, or the density of the bit 0, as their values are complementary.

We observe that we obtain $P - values$ greater than the threshold (sequences are random) for densities between $40 - 60\%$. In this situation 41000 inputs are classified as *non-random* (around 82%).

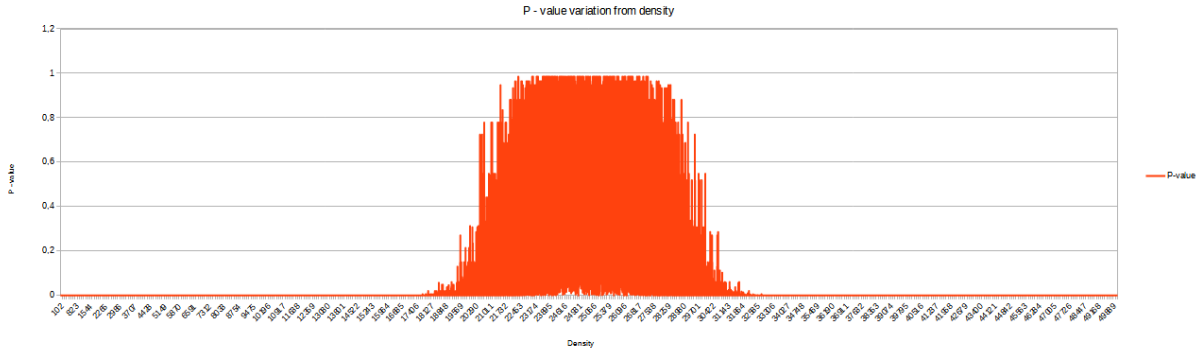

Figure 3: $P - value$ variation in respect to bits density (density iterations)

Figure 4 shows a more narrow frame of the data from Figure 3, for the interval of density $40 - 60\%$.

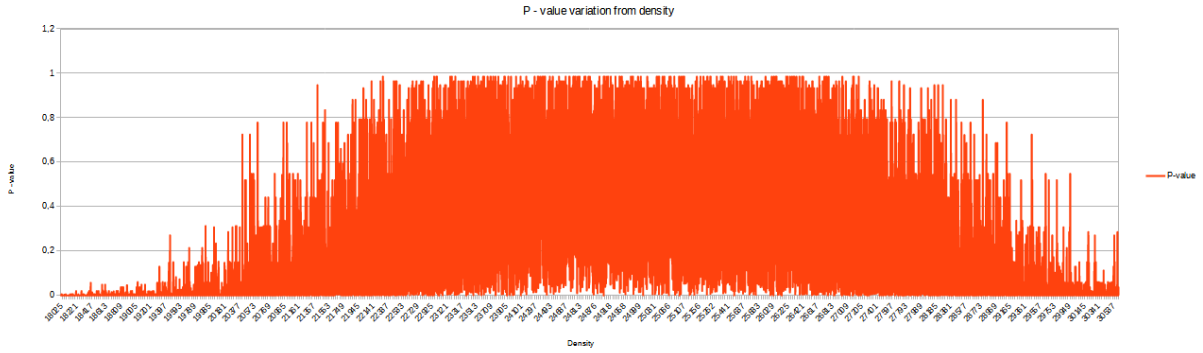

Figure 4: $P - value$ variation in respect to bits density (density iterations $40 - 60\%$)

After we did the above testing runs, there resulted $\approx$ 41000 inputs classified as *non-random* and $\approx$ 9000 inputs classified as *random* distributed over the density percentage as shown in Figure 5. We can observe that, for a density that tends to 50% we obtain a "peak" with more than 4000 random sequences - and almost no *non-random* sequences.

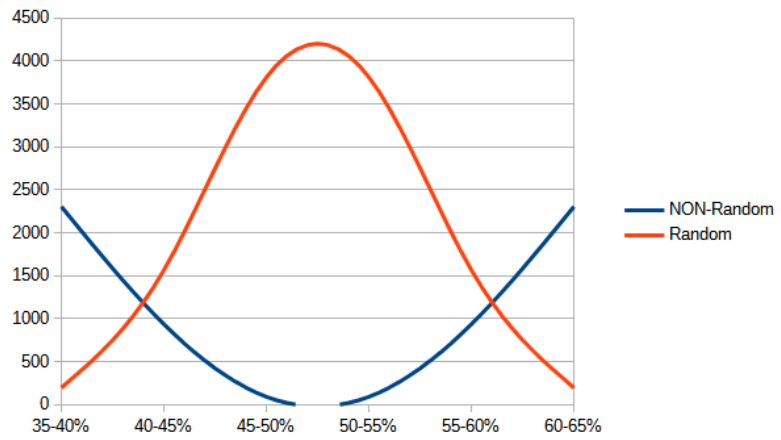

Figure 5: Random/Non-random count in respect to density percentage

4 Conclusions

In this study we analyzed one of the NIST Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, more exactly the Binary Matrix Rank Test.

Beside the statistical testing overview and the schema and the implementation of the test, we did some statistical experiments, then we made some remarks about the results. We concluded that a RNG or $PRNG$ should generate sequences having the bits density as closed as possible to 50% of the sequence length. For the 50000 length sample we generated, iterating all density possible values, we observed that, the more the value went far from the 50%, the more the number of random classified sequences decreased. We showed that it is feasible to predict the randomness of a bits sequence depending on how each bit's density is balanced.

As future work, we propose to do more experiments and analyze the correlation between the sequence randomness and the rank frequencies, F_M, F_{M-1} and $N - F_M - F_{M-1}$. Also, we want to calculate the relation between the truth of the null hypothesis and the test outcomes - and possible rerun all the previous test using standardized RNGs or $PRNG$s (i.e. 64-bit MELG [4], Squares RNG [5]).

References

[1] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J. and S. Vo, *"A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications"*, NIST 800-22, April 2010

[2] NIST standards: http://www.nist.gov/, http://www.csrc.nist.gov/

[3] Georgescu, C. and E. Simion, *"New Results Concerning The Power Of NIST Randomness Tests"*, Proceedings Of The Romanian Academy Series A Vol. 18, 2017, pp. 381-388

[4] Harase, S. and T. Kimoto, *"Implementing 64-bit Maximally Equidistributed F_2-Linear Generators with Mersenne Prime Period"*, 20 November 2017: https://arxiv.org/pdf/1505.06582.pdf

[5] B. Widynski, *"Squares: A Fast Counter-Based RNG"*, 25 October 2021: https://arxiv.org/pdf/2004.06278.pdf

[6] Neculache, N. and V. Petcu, *"NIST-BinaryMatrixRankTest"*, January 2022: https://github.com/Botoxit/NIST-BinaryMatrixRankTest