Journal of Computer Science 8 (8): 1353-1357, 2012
ISSN 1549-3636
© 2012 Science Publications

Testing Randomness: Poker Test with Hands of Three Numbers

[1]Wael Mohamed Fawaz Abdel-Rehim,
[2]Ismail Amr Ismail and [3]Ehab Morsy
[1]Department of Mathematics and Computer Science,
Faculty of Science, Suez Canal University, Suez, Egypt
[2]Department of Computer Science, College of Computers and Informatics,
Misr International University, Cairo, Egypt
[3]Department of Mathematics, Faculty of Science,
Suez Canal University, Ismailia 22541, Egypt

Abstract: Problem statement: The problem of testing randomness is motivated by the need to evaluate of the quality of different random number generators used by many practical applications including computer simulations, cryptography and communications industry. In particular, the quality of the randomness of the generated numbers affects the quality of such applications. In this study we focus on one of the most popular approaches for testing randomness, Poker test. Two versions of Poker test are known: the classical Poker test and the approximated Poker test, where the latter has been motivated by the difficulties involved in implementing the classical approach at the time it is designed. **Approach:** Given a sequence of n random numbers to be tested, the basic Poker approach divides this sequence into groups of k = 5 numbers, observes which of the possible patterns is matched by each quintuple, computes the occurring probability of each of these patterns and finally applies Chi-square test to check the randomness of such sequence. **Results:** For the sake of efficiency of the test, it is shownin the literature that, the value of k should be bounded from above based on the number of random numbers n to be tested On the other hand, most practical applications apply poker test with different values of k in order to ensure that the underlying sequence is truly random. This motivates implementing Poker approach with hands of three numbers in this study. **Conclusion:** We discuss the Poker test with hands of three numbers optimized for testing the randomness of sequences of sufficiently small sizes. From the computations point of view, we compare the performance of implementing Poker approach that uses hands of three, four, and five numbers and show that the running time of implementing the hands of three numbers is close to that hands of four numbers and is significantly less than that hands of five numbers.

Key words: Random numbers tests, cryptography, secret keys, poker test, approximated poker test

INTRODUCTION

Measuring the quality of randomness of a given sequence is a crucial problem that significantly affects the quality of many practical applications such as distributed algorithms, cryptography (Menezes *et al*., 1997), statistical sampling and computer simulation. In other words, the quality of such applications depends on generating unpredictable (random) sequence of quantities. From the practical point of view, such sequence must be of sufficiently large size in the sense that the probability of any particular value being selected must be sufficiently small in order to prevent an adversary from optimizing a search scheme based on such probability.

There are many techniques described in the literature for generating random and pseudorandom bits and numbers. A random bit generator is a device or an algorithm which outputs a sequence of independent and unbiased binary digits. A random bit generator can be used to generate uniformly distributed random numbers. However, generating of random bits is an inefficient procedure in most practical environments (storing and transmitting a large number of random bits are impractical if these are required in applications). We can overcome this difficulty by substituting a random bit generator with a Pseudorandom Bit Generator (PRBG); given a true random binary sequence of length k, PRBG is a deterministic

Corresponding Author: Wael Mohamed Fawaz Abdel-Rehim, Department of Mathematics and Computer Science, Faculty of Science, Suez Canal University, Suez, Egypt

algorithm that outputs a binary sequence of length l>> k which appears to be random. The main idea behind PRBG is to take a small truly random sequence and expand it to a sequence of much larger length in such a way that an adversary cannot distinguish between output sequences of the PRBG and random sequences of length l.

In order to make sure that such generators are secure enough, they should be subjected to a variety of statistical tests designed to detect the specific characteristics expected of random sequences. We now review a number of empirical tests described in the literatures; for further details (Kendall and Smith, 1938; Hamilton *et al*., 1997; Knuth, 1997; Sheskin, 1997).

Autocorrelation Test tests the correlation between numbers and compares the sample correlation to the expected correlation of zero.

Frequency Test develops frequency distribution of individual samples, uses the chi-square test to compare the distribution of the set of numbers generated to a uniform distribution.

Serial Test develops frequency distribution of pairs of samples. Then we compare the actual distribution against this expected distribution, using the chi-square test.

Gap test is used to examine the length of "gaps" between occurrences of samples in a certain range. It determines the length of consecutive subsequences with samples not in a specific range.

Runs Test tests the runs up and down or the runs above and below the mean by comparing the actual values to expected values. The statistic for comparison is the chi-square.

Poker Test (to be explained later in details) treats numbers grouped together as a poker's hand. Then the hands obtained are compared to what is expected using the chi-square test (Rutti, 2004; Stewart, 2009).

Note that, such techniques help detect certain kinds of weaknesses the generator may have by taking a sample output sequence of the generator and subjecting it to various statistical tests; each statistical test determines whether the sequence possesses a certain property that a truly random sequence would be likely to exhibit. That is, the conclusion of each test is not definite, but rather probabilistic. If the sequence is deemed to have failed any one of the statistical tests, the generator may be rejected as being non-random; alternatively, the generator may be subjected to further testing.

MATERIALS AND METHODS

Poker test: Here, we present in details the two versions of Poker test, the classical Poker test and the approximated Poker test.

Table 1: Different patterns of the classical Poker test of hands of 5 numbers and their probabilities

Name	Pattern	Probability
All different	ABCDE	0.3024
One Pair	AABCD	0.5040
Two pairs	AABBC	0.1080
Three of a kind	AAABC	0.0720
Full house	AAABB	0.0090
Four of a kind	AAAAB	0.0045
Five of a kind	AAAAA	0.0001

Classical poker test: The classical poker test consists of using all possible categories obtained from poker that uses hands of five numbers, i.e., AAAAA (five of a kind), AAAAB (four of a kind), AAABB (full house), AAABC (three of a kind), AABBC (two pairs), AABCD (one pair) and ABCDE (bust). In general, the poker test using hands of five numbers considers n groups of five successive integers denoted by (X_{5i}, X_{5i+1},…, X_{5i+4}), $0 \leq i \leq n$ and then observes which of the seven possible patterns is matched by each quintuple. The following Table 1 summarizes such patterns and their corresponding probabilities.

It is well known that the number of hands a poker test can apply with is not restricted to hands of five numbers (Kendall and Smith, 1938). In particular, Poker test that uses hands of four numbers is more convenient to be applied to certain applications such as simulation (Karian and Dudewicz, 1999) and cryptography (Brands and Gill, 1995; Menezes *et al*., 1997) in which we need to generate random integers or a random sequence of bits. For example, in cryptography, secret keys (used for encryption of messages or other purposes) are generated using Random Number Generators (RNGs) (Brands and Gill, 1995). Thus we want to apply Poker test to bit streams (typically represented by a 32-bit or 64-bit unsigned integer) rather than floating point numbers and since 64 bits is not evenly divisible by five we use the closest number that divides 64: 4. That is, the generated sequence of random numbers is divided into segments of four bits.

Given a sequence of n random numbers to be tested, it is shown that there is a limit based on n as to how large the value of k can be (Supaan, 2008). On the other hand, most practical applications apply poker test with different values of k in order to ensure that the underlying sequence is truly random (Fan *et al*., 2008). This motivates implementing Poker approach with hands of three numbers. In particular, for k = 3, we have the following categories: three of a kind (AAA), one pair (AAB) and a bust (ABC).

A Chi-square test is based on the number of quintuple in each category. We count the number of occurrences in each k-tuples and then use a chi-square analysis against the theoretical probabilities to

determine whether the stack represents a fair poker deck. The theoretical probabilities of such three categories (k = 3) can be computed in a similar way of that applied to the case of five (k = 4) and seven (k = 5) categories. For the sake of completeness, we compute such probabilities in details as follows. Clearly, the probability of choosing any number equals 1/10:

- The probability of choosing three of a kind = $\left(\frac{10}{10}\times\frac{1}{10}\times\frac{1}{10}\times\frac{3!}{3!0!}\right)=0.01$
- The probability of choosing one pair $=\left(\frac{10}{10}\times\frac{1}{10}\times\frac{9}{10}\times\frac{3!}{2!1!}\right)=0.27$
- The probability of choosing three of a kind $=\left(\frac{10}{10}\times\frac{9}{10}\times\frac{8}{10}\times\frac{3!}{3!}\right)=0.72$

Approximated poker test: At the time the classical Poker test is designed, checking the occurrences of these subsequences of length five using a computer program creates difficulties for the programmers as they have no one systematic similarity. In other words, the running time of such computations would be needed years using primitive computing machines. This motivates constructing a simpler version of the classical test to overcome the programming difficulties involved.

A good compromise would simply be to count the number of distinct values in the set of five (Knuth, 1997; Karian and Dudewicz, 1999). Namely, corresponding to the classical Poker test that uses hands of five numbers we get five categories, 1different, 2different, 3 different, 4 and 5 different. Thus, a finite time algorithms have been designed to implement such modified Poker test (Hamilton *et al*., 1997; Karl, 2008).

This breakdown is easier to determine systematically and the test is nearly as good. In general, we consider n groups of k successive numbers and then count the number of k-tuples with r different values. A chi-square test is then made using the following probability of the existence of r different Eq. 1:

$$\Pr=\frac{d(d-1)...(d-r+1)}{d^k}\left\{\begin{matrix}k\\r\end{matrix}\right\} \quad (1)$$

where, $\left\{\begin{matrix}k\\r\end{matrix}\right\}$ denote the Stirling number of the second kind (the number of ways to partition a set of k elements into exactly r parts). The Stirling number can be computed using a well known formula. For example, the values for the Stirling numbers for k = 3 and r = 1, 2, 3, are 1, 3, 1, respectively.

The classical Poker test with hands of three numbers attains a corresponding approximated version based on Stirling number by considering only three categories, 1, 2 and 3 different. To calculate the expected values we use equation 1 with d = 10. Now, we determine theoretical probabilities of such categories:

$$\Pr(1\text{ different})=\frac{10}{10^3}\left\{\begin{matrix}3\\1\end{matrix}\right\}=0.01$$

$$\Pr(2\text{ different})=\frac{10(10-1)}{10^3}\left\{\begin{matrix}3\\2\end{matrix}\right\}=0.27$$

$$\Pr(3\text{ different})=\frac{10(10-1)(10-2)}{10^3}\left\{\begin{matrix}3\\3\end{matrix}\right\}=0.72$$

Then different hands obtained can be compared to what is expected using the chi-square test to see how far the data has strayed from the theoretical distribution.

Note that the two version of the Poker test are identical in the case of using hands of three numbers.

RESULTS AND DISCUSSION

We now implement and compare the running time of the classical Poker test that uses hands of three, four, and five numbers. We implement programs using C++ code that create random numbers and count the occurrence of these differences or count number of occurrences, then classified each to possible type of poker hand. Finally, it determines the chi-square. The experimental results are reported on PC 2.4 GHz, 1024 MB of RAM, 256 KB of cache.

Example: We implement Poker test with hands of three numbers on the one million digits. The degrees of freedom (df) for chi-square table equals 2 (Table 2). If the computed value of chi-square is equal to or greater than the tabled critical value at the prespecified level of significance, then the null hypothesis is rejected and hence the distribution is not truly random. Otherwise (the computed value of chi-square is less than the tabled critical value), the null hypothesis is retained. That is, the data is consistent with the series being random.

For df = 2, we get $X^2_{.05}$ = 5.99 and $X^2_{.01}$ = 9.21. Since the obtained value X^2 = 3.35 is less than $X^2_{.05}$ = 5.99, the null hypothesis is retained. This implies that the underlying data is truly random.

Now, we analyze Chi-Square for both the classical and the modified Poker test approaches described in Fig. 1.

1. Read number of hands to deal
2. Open File to read random numbers
3. do
4. Loads the hands into an array // a sequence of 3 numbers
5. Determine which kind of combination this group of 3 contains
6. Count the number of similar values; breaks at 3
7. Increment the appropriate counter //(3 counters: all different, one pair and three of a kind)
8. While (loads the hands < number of hands)
9. Calculate the percentage of the n total repetitions corresponding to each counter
10. Computes the expected theoretical values
11. Compute chi square using the expected probabilities
12. Measure execution time in the program
13. Print "The program execution rime"
14. Print "Chi square"

Fig. 1: The classical poker test with hands of three numbers pseudo-code algorithm

Table 2: Chi-Square analysis for poker test with hands of 3 numbers

Cell/Poker Hand	Observed number of hands (O)	Expected number of hands (E)	$\frac{(O-E)^2}{E}$
Busts (All different)	240474	240000	0.1365
One pair	89547	90000	2.2792
Three of a kind	3312	3333	0.9371
Sums	333333	333333	$X^2 = 3.35$

We apply the three methods to ensemble of different size and checked the results to see if they are within a specified confidence level. The results are shown in the following Table 3.

We observe that df = 6, df = 4 and df = 2 respectively for the classical poker test with hands of 5, 4 and 3 numbers respectively. Our chi-squared values is less than the critical value for the 0.05 significance level (12.9 to be precise in the case of hands of 5 numbers, 9.49 in the case of hands of 4 numbers and 9.49 in the case of hands of 3 numbers), we accept the null hypothesis as true and conclude that the two methods seemed to produce acceptable chi-square statistics. The chi-squares were within the 95% confidence interval.

Finally, we analyze the running time of the classical Poker test with hands of 5, 4 and 3 numbers described in Fig. 1. We determine the running time of all these approaches in milliseconds. The resulting running time for the classical approaches is shown in the following Table 4.

The results of Table 4 (shown in Fig. 2) imply there is a significant improvement in term of the running time in the case of applying the classical Poker test with hands of 3 and 4 numbers, especially when the number of random numbers is sufficiently large.

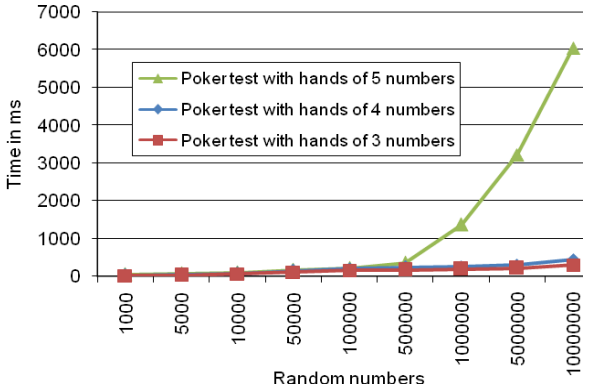

Fig. 2: Performance comparison of the classical poker approaches with hands of 3 and 4 numbers in term of their implementing times

Table 3: Chi-Square analysis for Poker test with hands of 3, 4, and 5 numbers

No. of random Numbers	Poker test with hands of 3 num. Chi-Square Value	Poker test with hands of 4 num. Chi-Square value	Poker test with hands of 5 num. Chi-Square value
1000	0.75	4.29	7.93
5000	0.88	1.91	7.10
10000	1.45	6.43	3.42
50000	2.63	2.09	3.63
100000	2.42	1.71	3.51
500000	3.16	5.34	7.64
1000000	3.35	5.49	2.63
5000000	1.53	4.77	1.65
10000000	1.10	6.20	3.37

Table 4: Running time for Poker test with hands of 3, 4, and 5 numbers

No. of random Numbers	Poker test with hands of 3 num.	Poker test of with hands of 4 num.	Poker test of with hands of 5 num.
1000	16	32	47
5000	31	47	78
10000	67	79	93
50000	109	140	172
100000	156	204	219
500000	172	234	359
1000000	188	256	1375
5000000	219	297	3219
10000000	296	437	6047

CONCLUSION

We have been studied Poker test, one of the most popular approaches for testing randomness. In particular, we have been discussed the Poker test with hands of three numbers optimized for testing the randomness of sequences of sufficiently small sizes. From the computations point of view, we have been compared the performance of implementing Poker approach that uses hands of three, four, and five numbers and have been shown that the running time of

implementing the hands of three numbers is close to that of hands of four numbers and is significantly less than that of hands of five numbers.

REFERENCES

Brands, S. and R. Gill, 1995. Cryptography, statistics and pseudo-randomness. I. Prob. Math. Stat., 15: 101-114.

Fan, Z., X. Tian, J. Song and X. Li, 2008. Pseudo-random sequence generator based on the generalized Henon map. J. China Univ. Posts Telecommun., 15: 64-68. DOI: 10.1016/S1005-8885(08)60109-0

Hamilton, J.A., D.A. Nash and U.W. Pooch, 1997. Distributed Simulation. 1st Edn., CRC Press, Boca Raton, ISBN-10: 0849325900, pp: 390.

Karian, Z.A. and E.J. Dudewicz, 1999. Modern Statistical, Systems, and Gpss Simulation. 2nd Edn., CRC Press, Boca Raton, ISBN-10: 0849339227, pp: 527.

Karl, A., 2008. Pseudorandom numbers: Generation, statistical measures, monte carlo methods and implementation in C++. Senior Thesis, Department of Mathematics, University of Notre Dame.

Kendall, M.G. and B.B. Smith, 1938. Randomness and random sampling numbers. J. Royal Stat. Soc., 101: 147-166. DOI: 10.2307/2980655

Knuth, D.E., 1997. The Art of Computer Programming. 3rd Edn., Addison-Wesley, Mass, ISBN-10: 0201896834.

Menezes, A.J., P.C. Van Oorschot and S.A. Vanstone, 1997. Handbook of Applied Cryptography. 1st Edn., CRC Press, Boca Raton, ISBN-10: 0849385237, pp: 780.

Rutti, M., 2004. A random number generator test suite for the C++ standard. Diploma Thesis, Institute for Theoretical Physics, ETH Zurich.

Sheskin, D., 1997. Handbook of Parametric and Nonparametric Statistical Procedures. 2nd Edn., CRC Press, Boca Raton, ISBN-10: 0849331196, pp: 719.

Stewart, W.J., 2009. Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. 1st Edn., Princeton University Press, Princeton, ISBN-10: 0691140626, pp: 758.

Supaan, S.B., 2008. Analysis for A5/1 and A5/2 algorithm (stream ciphers). Senior Thesis, Faculty of Electrical Engineering, Universiti Teknologi Malaysia.