Trends in Applied Sciences Research

ISSN 1819-3579

Academic
Journals Inc.

www.academicjournals.com

Trends in Applied Sciences Research 10 (5): 231-244, 2015
ISSN 1819-3579 / DOI: 10.3923/tasr.2015.231.244

A General Evaluation Pattern for Pseudo Random Number Generators

[1]Ahmad Gaeini, [1]Abdolrasoul Mirghadri and [2]Gholamreza Jandaghi
[1]Imam Husein Comprehensive University, Iran
[2]Faculty of Management and Accounting, Farabi College, University of Tehran, Iran

Corresponding Author: Gholamreza Jandaghi, Faculty of Management and Accounting, Farabi College, University of Tehran, Iran

ABSTRACT

Many models have been designed for generating pseudo random numbers and there is the potentiality to design more. Among the mentioned generators, a few of them are applied in cryptography. Thus, there is an inevitable need among researchers and users for evaluating the generators. A variety of criteria are used for evaluation which have more or less or neutral importance depending on the way the generators are applied. Accordingly, some reliable sources pay attention to the particular criterion while overlooking others. In the present study, in addition to enumerating all of the popular criteria, there has been an attempt to classify them in the form of a general model. At the end, some suggestions are presented for the foremost priority in using and applying the criteria. With implementing this pattern all the criterions have been considered and on the other hand the probability of second type error will be reduced. Also, observance of the sequence of criterions would cause saving in time and costs.

Key words: General evaluative model, pseudo random number generators, cryptography

INTRODUCTION

Random numbers are applied in several sciences such as simulation, modeling, computer sciences, statistical sampling and cryptography. With the expansion of network communications and unauthorized accessibility to the exchanged information, the development of research seems necessary. Random numbers play such a key role in cryptography that it seems impossible to consider a cryptography program without using random numbers (Kang, 2005). Using insecure random numbers in cryptographic systems leads to the insecurity of the whole system. Thus, evaluating the generators and not using insecure generators is a significant step in achieving secure cryptographic systems.

In the following, some applications of random numbers in cryptography are mentioned:

- Message and session keys for symmetric ciphers, such as triple-DES or Blowfish
- Seeds which usually produce mathematical values, such as prime numbers in RSA or ElGamal
- Salts to combining with the passwords which cancel the program for guessing the passwords in offline mode
- Initialization vectors for chaining modes
- Random values for special requests from digital signature schemes, such as DSA

- Random challenges in authentication protocols, such as Kerberos
- Nonces for protocols, to ensure that different runs of the same protocol are unique; e.g., SET and SSL (Kelsey *et al*., 1999)

Generating true random numbers based on physical sources by computers is time consuming and costly. The production speed is usually low in these generators, thus, pseudo random number generators are used which are able change a short sequence of random numbers into a long sequence of numbers in a way that appear to be random. "Indistinguishable from truly random sequence" and "unpredictability to an adversary with limited computational resources" are two main requirements for output of pseudo random number generators (Kang, 2005).

Confirming the fact that a generator is really pseudo random is difficult but one can reject that a generator is pseudo random by statistical tests and the mentioned essentials. Therefore, success in a statistical test can be regarded as a criterion for identifying suitable generators.

Given that the method for distinguishing between output sequences of generators and true random sequences is called attack (Kelsey *et al*., 1998), an appropriate generator has to be resistant to identifiable attack. Then, resistance against identified attacks is another criterion for identifying an appropriate generator. Other factors such as speed, long sequence can be used as a criterion for identifying a good generator.

Patidar *et al*. (2009) run NIST (3 level) and DIEHARD tests for their generators based on formed chaos standard mappings in. They took this as the only criterion for analyzing generators (Patidar and Sud, 2009). Orue *et al*. (2010) proposed a generator based on Fibonacci mapping and evaluated that in terms of speed, easy implementation, long period and success in statistical tests. Gayakwad and Ranbir (2012) proposed a useful pseudo random number generator for processing image. They evaluated the generator with criterion such as speed, easy implementation, Monobit and serial tests and other NIST tests.

Akhshani *et al*. (2014) for pseudo random number generator based on quantum chaos mapping applied 4 famous souit of statistical tests, DIEHARD, ENT, NIST and U0I. They also considered speed and easy implementation criterions.

Reyad and Kotulski (2015) presented a generator that is combination of elliptic curves and chaos mapping and for its evaluation, they investigated only 5 statistical tests addition to speed and long period criterions.

In this study, there is an attempt to present all the criteria proposed by researchers in cryptography as an applicable evaluative model so that one can evaluate his/her generator or other generators without referring to several sources. Then they can use the criteria based on the priority we proposed.

James (1990) enumerated the criteria for a suitable generator as long period, uniform distribution, repeatability, having disjoint subsequences and portability. Kelsey *et al*. (1999) proposed the comprehensive classification of several attacks on the generator. Kelsey *et al*. (1998) suggested four criteria as resistance against attacks, efficiency, the capacity for reseeding and resistance against disavowal. Given the significant role of chaos mapping designing pseudo random number generators, paying attention to essentials of chaos based systems proposed by Gonzalo and Li (2006) would be necessary. Statistical packages are a set of statistical tests which, in designers' perspective, can be a criterion for the suitability of generators if they are not rejected. Applying programs such as DIEHARD in reference (Marsaglia, 1996), ENT in reference (Walker, 2008), U0I

in reference (L'Ecuyer and Simard, 2007) and NIST in reference (Rukhin *et al*., 2010) are available. Babaei and Farhadi (2011) suggested that the essentials for a pseudo random number generator are uniform distribution, independence, the long period and unpredictability and offered the use of U0I package for examining these essentials (Babaei and Farhadi, 2011). Yang and Xiao-Jun (2012) applied autocorrelation and NIST tests for the suggestive generators based on empirical distribution of data. Furthermore, they examined the analysis of the key space and sensitivity and speed analysis (Yang and Xiao-Jun, 2012). Francois *et al*. (2013) applied the NIST tests for three types of output sequences and made a correlation analysis for three subsequences and examined the security analysis for the size of the key space, key sensitivity and key selection. Then, they studied the resistance against guess-and- determine attacks (Francois *et al*., 2013). Ilyas *et al*. (2013) applied for a suggestive generator the NIST package in deeper level which was a uniform distribution on p-value (Ilyas *et al*., 2013). Stoyanov and Kordov (2014) applied three testing packages namely, DIEHARD, ENT and NIST on a large number of a generator based on chaos. Then, he examined the portion of successful sequences (Stoyanov and Kordov, 2014). Wang and Nicol (2015) showing the weak points of NIST package with respect to indicating nonrandom generators as random, suggested that there should be tests based on statistical distances such as iterated logarithm for decreasing type two error (Wang and Nicol, 2015). Hamdi *et al*. (2015) proposed the criteria for long sequences and uniform distributions for generators and examined the speed and resistance analysis against differential attacks. Moreover, the NIST tests were applied in several output sequences and the results were compared based on the ratio of success in AES algorithm (Hamdi *et al*., 2015).

It should be noted that, according to the current trend, in order to evaluate the suggestive generators, only a few criteria can be considered to accept a generator while the same generator may be rejected if evaluated by other criteria. In this study, considering other criteria proposed in the literature and based on the experiences of the authors, a model is proposed which has a general perspective toward evaluating random number generators. Observing this model causes increase in security level of generators and thus, increase in security of cryptographic systems.

MATERIALS AND METHODS

General definition and concepts: As a basic definition of pseudo random number generators, it is necessary, first, to elaborate on the definition of indistinguishability of two ensembles (Goldreich, 2004).

Definitions 1: Two ensembles $\{X_n\}_{n\in N}$, $\{Y_n\}_{n\in N}$ are called indistinguishable in polynomial-time when we have for each probabilistic polynomial-time algorithm A and each polynomial p and all sufficiently large n,s:

$$\left|\Pr\left[A\left(X_n\right)=1\right]-\Pr\left[A\left(Y_n\right)=1\right]\right|<\frac{1}{p(n)} \tag{1}$$

Ensemble $\{X_n\}_{n\in N}$ would be pseudo random if $\{X_n\}_{n\in N}$ and $\{U_n\}_{n\in N}$ be indistinguishable, which U_n shows that the random variables have uniform distribution on binary sequence set with the length of n.

Definition 2: Deterministic polynomial-time algorithm G are called pseudo random number generator if the two conditions will be met (Goldreich, 2004):

- The function l exists that $l(n)>n;\forall n\in N$ and for each $s\in\{0,1\}^*$, we will have: $|G(s)| = l(|s|)$, where s is called seed
- Ensemble $\{G(U_n)_{n\in N}\}$ be pseudo random

Definition 3: Ensemble $\{X_n\}_{n\in N}$ is called unpredictable in polynomial-time, if for each probabilistic polynomial-time algorithm A and each polynomial p and all sufficiently large n,s (Goldreich, 2004):

$$\Pr\left[A\left(X_n\right) = \text{next}_A\left(X_n\right)\right] < \frac{1}{2} + \frac{1}{p(n)} \tag{2}$$

The above definitions are not practically possible since all the algorithms should be confirmed. In the following definitions there seems to be a good association between pseudo random number generators and statistical tests (Goldreich *et al*., 1986).

Definition 4: It is said that a pseudo random number generator to pass all the polynomial-time statistical tests if no polynomial-time algorithm can correctly distinguish between an output sequence of a generator and the truly random sequence with the same length with probability significantly greater than $\frac{1}{2}$.

In other words, the output of the generator and the truly random sequence with the same length are computational indistinguishable.

In reference (Goldreich *et al*., 1986), it is confirmed a generator can be successful in an unpredictable next bit if and only if becomes successful in all polynomial-time statistical tests.

Lack of success of a generator in any statistical test could be a reason for rejecting that generator. Although the success of a generator in all of the existing tests could be regarded as a strength point, however, it cannot be considered as a reason for their being pseudo number.

Evaluating model for pseudo random number generators: In this section, a model for evaluating pseudo random number generator will be proposed. Investigating theoretical underpinnings and common methods researches applied during recent years leads to a conclusion that for general evaluation of a pseudo random number generator, actions must be done in three dimensions. In fact, a suitable generator should have the following features:

- In applying statistical tests on generators, the random and independence assumption of the output should not be rejected
- Being resistant against identifiable and appropriate attacks
- Being acceptable with respect to applying and security issues

Accordingly, three subsections will be proposed.

Statistical tests: We can determine whether an output sequence is nonrandom, however, we cannot confirm that a sequence is random. In statistical tests, we can decide with a determined

measurable probability about whether a generator produces random sequence. Null hypothesis suggests that the output of a generator is random, that is, it follows independently uniform distribution or i.i.d. Type one error occurs in a case that we reject the random sequence $\alpha = P$ (rejected $H_0 | H_0$ true). According to sample observations, p-value will be calculated and if p-value<α, the null hypothesis will be rejected and otherwise, the hypothesis will be accepted. The success of a generator lies in the fact that the H_0 will be accepted. Statistical tests could be conducted in three levels. The first level selects an output sequence from the generator and the test will be conducted on it. The second level considers m sequences which have n number of bits as sample of output generators and each test will be conducted on all of sequences. Given that m independent test are conducted in α level, H_{0i} shows random assumption of ith sequence. Random variable Bernoulli X_i will be 1 when H_{0i} is accepted. Thus, the variable:

$$Y = \sum_{i=1}^{m} X_i$$

has polynomial distribution with m and 1-α parameters, when each H_{0i} is true. Now, if we indicate the probability of acceptance of the generator with $\gamma = P(X_i = 1)$, then the variable:

$$Y = \sum_{i=1}^{m} X_i$$

has polynomial distribution with m and γ parameters. Now, for second test, the hypothesis H_0: $\gamma \geq 1-\alpha$ has to be tested in a significant α* level. The test statistic, if the null hypothesis is true, has normal limiting distribution and is shown as following:

$$Z = \frac{Y - m(1-\alpha)}{\sqrt{m\alpha(1-\alpha)}} \tag{3}$$

Null hypothesis will be rejected if p value<α* and otherwise, we can accept that the generator is pseudo random.

The acceptance region for this test shows the minimum acceptable ratio for sequences:

$$(1-\alpha) - Z_{\alpha^*}\sqrt{\frac{\alpha(1-\alpha)}{m}} \tag{4}$$

For example, in examining m = 1000 sequences for α = 0.01, if the second level test for α* = 0.0001 was applied, we should have at least 979 sequences to accept the fact that the generator was pseudo random.

The third level is based on the following lemma.

Lemma 1: If a test statistic were continuous, p value as a random variable has uniform distribution on [0,1], if the primary hypothesis is true.

Proof: Let $a \in (0,1)$ because of continuity in test statistic T, ta number will be:

$$Pr(T \leq ta) = a$$

Now, if we show the random variable p value with p, two inequality $p \leq a$ and $T \leq ta$ indicate the rejection of H_0 with the probability of a, so they are equivalent. $Pr(T \leq ta) = Pr(P \leq ta) = a$, so according to last equality and uniform distribution function, problem has been proofed.

With regards to lemma 1, we can conduct the first level of test for several times and consider p-value as a random sample of p. Then, by using goodness of fit tests such as Kolmogorov-Smirnov test and Chi-squared test, we examine the establishment of uniform distribution for p.

It should be noted that for conducting statistical tests, there are identified packages with different capacities and applications. Examining all of these packages we could find out that the package which belongs to NIST is more appropriate than others. Provided that the package has the essential standards, we can introduce a sequence which is obviously nonrandom in a sense that passes all the tests successfully (Wang and Nicol, 2015). The iterated logarithm test which is based on the Law of the Iterated Logarithm (LIL) is a tool for reducing the potential type two errors. In fact, a generator which is confirmed by this test, it will be less likely to produce nonrandom sequences. This test is called the fourth level for statistical tests.

Iterated logarithmic test: Two famous limit theorems about binary sequences are central limit theorem and law of the iterated logarithm. A number of NIST tests are based on central limit theorem but the law of the iterated logarithm has never been used in any tests.

Definition 5: It is said that the sequence ξ does not pass the weak LIL-(α,N*) test successfully if for any $n \in N$ we have:

$$-1+\alpha < S_{lil}(\xi|n) < 1-\alpha \quad (5)$$

Where:

$$S_{lil}(\xi|n) = \frac{2\sum_{i=0}^{n-1}\xi[i] - n}{\sqrt{2n \ln \ln n}}$$

and $\xi[i]$ is the i-th bit of ξ. Further, $\alpha \in (0,0.25)$ and $N^* \subseteq N$.

Common attacks on pseudo random number generators: Generally, a pseudo random number generator involves an unpredictable input called seed value and a secret state "S". The pseudo random number generator starts with a random state and has to process many seeds to reach a secure state S. The output should be produced which an attacker S cannot guess. Moreover, the generator should be able to change its secret state by repeatedly processing inputs (seed). Figure 1 describes the pseudo random number generators (Kelsey *et al*., 1998).

One of the features that makes the use of pseudo random number generators in cryptography distinguished from others is identifying the attacks on these generators and examining the resistance against them. In an overall view, attack is defined as any method of distinguishing between pseudo random number generator outputs and truly random outputs (Kelsey *et al*., 1998). Nevertheless any attempt in achieving the following goals can be regarded as an attack:

- To learn about the output
- Getting information with respect to input and output
- To manipulate the output

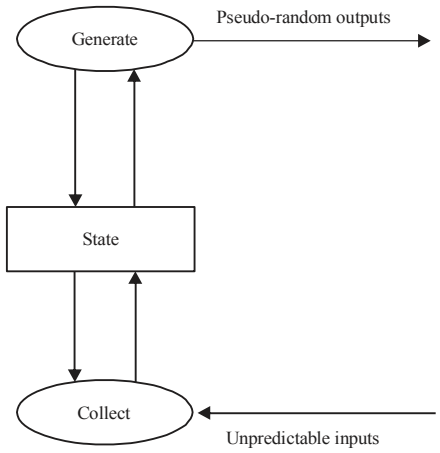

Fig. 1: PRNG model (Kelsey *et al*., 1998)

Based on the above definition, a wide range of attacks are identified. The following classification is done based on the common identified attacks (Kelsey *et al*., 1998).

Direct cryptanalytic attack: When an attacker can directly distinguish between output sequences of generator and truly random number sequences, a direct cryptanalytic attack has occurred. Distinguishing attack is regarded as this type of attack. A large number of generators apply cryptographic primitives such as hash functions or block ciphers in order to prevent such attacks.

Input-based attacks: When the attacker can observe the input or manipulate (control) it, an input-based attack occurs. The goal of the attacker is to reduce the number of outputs in such a way to make it easy to guess them. In order to prevent this type of attack, it is better to transform the inputs with an enumerator into a hash function, before transferring them to the generator. The guess-and-determine and differential attacks are considered as this type of attack.

State compromise extension attacks: When the attacker is able to recover unknown pseudo random number generator outputs from before S was compromised, this attack occurs. The attacker tries to develop this knowledge to further points in time and to previous or future output. This attack usually occurs when the generator starts with insecure mode such as the primary which has all the elements zero or the selection of seed was from a set available to the attacker. In order to resist against this attack, enough care must be taken for selecting seed. Meet-in-the-middle attacks are considered as this type of attack.

Other criteria: In this section, we discussed the criteria which are not examined in statistical test and cannot be regarded as attacks except for one.

Speed: The expression 'it is easy to design a secure but very slow cipher' (Alvarez and Li, 2006) is well known in cryptography. Thus, attention to speed in evaluating the generators is something common. Speed becomes significant in proportions because factors such as the structure of CPU, the memory capacity, platform, optimum coding and programming language affect the speed. For chaos-based generators, the expected speed should not be slower than 10 Mbps on a 1 GHz-CPU.

Effective implementation: The expression "It is quite easy to design a secure but very large cipher" (Gonzalo and Li, 2006) is common in cryptography. Thus in evaluating generators, it is required that we pay attention to the fact that it is better to have a simpler generator and less need to hardware and software equipments.

Key space: In designing generators, usually it is possible to call the used seeds or initial vectors as "key". In current conditions the size of key space for resisting against brute-force attacks should be at least 2^{128}. The bigness of the size of key space is necessary but not enough because, in cryptographic perspective, the keys should not be strong and correlated (Francois *et al*., 2013).

Key sensitivity: The sensitivity on the key is an essential doer for chaos-based generators. The key sensitivity can be calculated by correlation the outputs which are the result of seeds that are close to each other. Insignificance of correlation shows high sensitivity. If the test was rejected, then the correctness of the null hypothesis could be questioned, that is, the generator is not pseudo random.

Suggested priorities: Applying a variety of criteria is a strength point for the model of evaluating pseudo random number generators. However, it should be noted that not caring about appropriate priorities could lead to waste of time and deviation from the main goal.

Thus, the priorities are empirically suggested by comprehensively examining the research conducted on studying the association and correlation of criteria.

In the following model, if the generator can meet the condition in each level, goes to next level. Otherwise, evaluation procedure is stopped and the generator is recognized inappropriate for cryptography points. However, it should be noted that inappropriate recognized generators may be used for certain applications in special cases.

First step

Speed: The speed of production should be measured for the evaluating generator. If the calculated speed was more than the criterion mentioned by Alvarez and Li(2006), we could continue the evaluation, otherwise, we will reject the generator.

Second step

Resistance against the brute-force attack: We measure the size of the key space and if the criterion were more than 2^{128}, we go to the next step in evaluation and, otherwise, the generator will be rejected.

Third step

Implementation of statistical tests in the first level NIST: Given to the fact that in some tests the number of elements under evaluation should be 10^6, thus, we produce a sequence with the minimum number of 10^6 members of the generator and then implement the first level NIST on the sequence. If random assumption were not rejected with any of the tests, we go to the next step in evaluation, otherwise, we reject the generator.

Fourth step

Examining the sensitivity: In order to examine the sensitivity of the generator to the key and initial values, close seeds are selected and we get the output. Then, the correlation of output sequences will be calculated. Insignificance of correlation between the outputs means that the

sensitivity is to the key is high and the generator will be accepted. Then we go for evaluation to the next step. In the case of significant correlation between the generators, the generator will be rejected.

Fifth step

Examining the resistance against attacks: Given to the mentioned attacks in Kelsey *et al*. (1998) and the structure of the generator, the resistance against the attacks will be examined and in the case of success, each generator is rejected. Is the generator were resistant against the known attacks, we go to the next step in evaluation.

Sixth step

Implementing statistical tests second level NIST: A large number of sequences of generator were selected as a random sample and these tests will be conducted independently. The ratio of the accepted sequences is compared by Eq. 4. If for all the NIST tests our ratio was bigger, the generator is accepted and we go to the next step in evaluation.

Seventh step

Implementing statistical tests the third level NIST: A large number of sequences are selected from the generator as samples and each test will be independently conducted on them. For each test a sequence of p value is calculated. Uniform distribution of random variable p is tested and we can reject the generator if it was rejected.

Eighth step

Implementing the low of iterated logarithm test: Running this test is applied in following stages (Wang and Nicol, 2015):

- Select testing points such as $n = 2^{t+26}$ that t = 0, 1,..., 8
- Generate $m \geq 10000$ sequences and put them in $\mathbb{R}$ set
- Compare the distance between $\mu_n^U, \mu_n^{\mathbb{R}}$ for testing points of first stage. If the distance was negligible, accept the generator. For simplicity, let the discrete partition Ω as following set for the set of real numbers and use these definitions:

$$\{(-\infty,1),[1,\infty)\}\cup\{[0.05x\text{-}1,\ 0.05x\text{-}0.95):\ 0\leq x\leq 39\}$$

$\mu_n^U\left((-\infty,x]\right)\cong\Phi\left(x\sqrt{2\ln\ln n}\right)$, that Φ is the standard normal cumulative distribution function and $\mu_n^{\mathbb{R}}(I)=\Pr\left[S_{lil}(x)\in I, x\in\mathbb{R}\right]$, that I is any Lebesgue measurable set on R.

Also, for calculating distance this statement, root-mean-square deviation, can be used:

$$\mathrm{RMSD}\left(\mu_n^{\mathbb{R}},\mu_n^U\right)=\sqrt{\frac{\sum_{A\in\Omega}\left(\mu_n^{\mathbb{R}}(A)-\mu_n^U(A)\right)^2}{|\Omega|}} \tag{6}$$

As Fig. 2 shows, this pattern neither puts the attack criteria like many recourses (Kelsey *et al*., 1998, 1999), nor pays no attention to attack like many other recourses (Reyad and Kotulski, 2015; Babaei and Farhadi, 2011; Ilyas *et al*., 2013; Hamdi *et al*., 2015). Also, about statistical tests intended by all the researchers, our proposed model completes recourses perspectives only deal with tests (Patidar and Sud, 2009; Ilyas *et al*., 2013; Hamdi *et al*., 2015). Furthermore, the

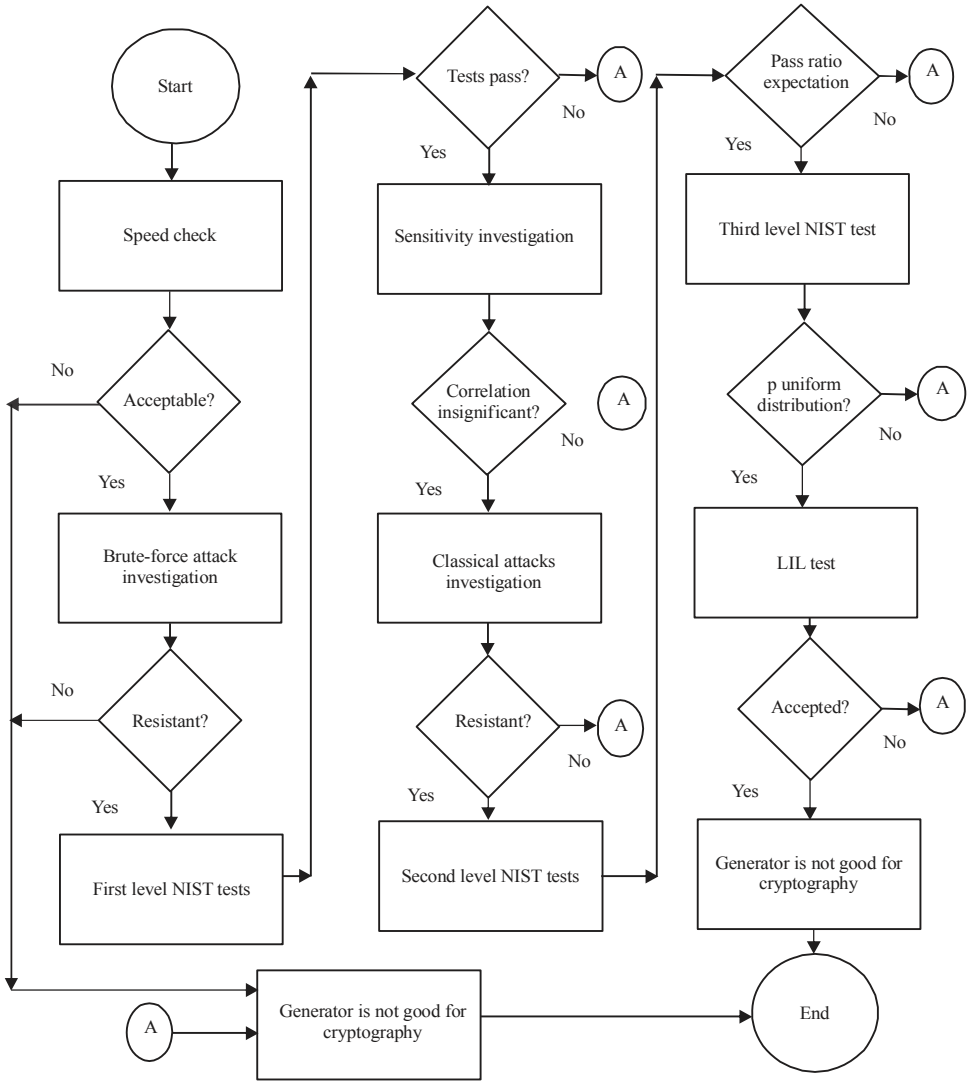

Fig. 2: PRNGs evaluation pattern

recourses that only have noted first level of tests can have a precise evaluation too (Walker, 2008; Babaei and Farhadi, 2011; Ilyas *et al*., 2013), so this pattern completes their evaluation. Because of inappropriate generators that success in all 3 levels, as mentioned in recourse (Wang and Nicol, 2015), implementing iterated logarithm test is an important step for reducing second type error probability that our proposed pattern has focused on it.

RESULTS AND DISCUSSION

First in this section, we presented how generators stop or pass steps of the pattern with considering various generators. Then a generator will be proposed that can pass most of the steps.

Examples for some steps: For first step 3 generators are evaluated and results are shown in following table (Table 1). As it is presented in Table 1, Yang's generator does not have acceptable speed. So, it is rejected and there is no need to run other steps.

For second step some generators are evaluated and the results shown in Table 2. As it is indicated in Table 2, first and second generators are rejected because of their small size of key space (complexity) and there is no need to continue pattern for them.

For third step sensitivity of generators are evaluated and results are shown in Table 3. As it is shown in this table, second generator can't pass this step and there is no need to run the other steps.

For sixth and seventh step, BBS generator can be used which passes the sixth step successfully but stops at seventh step. It is indicated about some NIST tests in following table (Table 4).

For eighth step some generators have been evaluated and just 2 generators could pass the LIL test successfully. Table 5, indicates results for this step.

Introducing a good generator: Francois and Defour (2013) proposed a pseudo random generator with using 3 logistic mapping chaos in 2013. The main idea of their PRBG is to combine several

Table 1: Table speed for speed check

Generators	Speed (Mbps)	Results
L. Yang (Yang and Jun, 2012)	0.4844	Rejected
G. Alvarez (Alvarez and Li, 2006)	40	Accepted
M. Francois (Francisco and Defur, 2013)	44.112	Accepted

Table 2: Complexity for brute force attack

Generators	Complexity	Results
A5/1	2^{56}	Rejected
DES	2^{56}	Rejected
3DES	2^{168}	Accepted
AES256	2^{256}	Accepted

Table 3: Result of NIST test on the sequences obtained from the two generators

	M. francois (Francisco and Defur 2013)		V. patidar (Patidar *et al*., 2009)	
Test names	p-value	Results	p-value	Results
Frequency	0.888272	Accepted	0.218594	Accepted
Block-frequency	0.013966	Accepted	1.000000	Accepted
Cumulative sums (1)	0.851269	Accepted	0.343496	Accepted
Cumulative sums (2)	0.723802	Accepted	0.284913	Accepted
Runs	0.239428	Accepted	0.000000	Rejected
Longest run	0.341867	Accepted	0.000000	Rejected
Rank	0.690933	Accepted	0.611764	Accepted
FFT	0.704824	Accepted	0.000000	Rejected
Non-overlapping	0.014372	Accepted	0.000000	Rejected
Overlapping	0.544746	Accepted	0.000000	Rejected
Universal	0.693543	Accepted	0.000000	Rejected
Approximate entropy	0.534042	Accepted	0.000000	Rejected
Random excursions	0.016321	Accepted	0.013588	Rejected
Random Ex-variant	0.014383	Accepted	0.125754	Accepted
Serial (1)	0.532881	Accepted	0.000000	Rejected
Serial (2)	0.508815	Accepted	0.000000	Rejected
Linear complexity	0.956706	Accepted	0.817657	Accepted

Table 4: Status for proportion of passing and distribution pattern

Parameters	Test 12	Test 13	Test 14
Expected ratio	0.972766	0.977814	0.983907
Observed ratio	0.983333	0.991667	0.986667
Result	Success	Success	Success
p-value of p-values (POP)	9.157321e-01	8.425709e-07	2.226391e-01
Result	Uniform	Uniform	Non-uniform

Table 5: Law of the iterated logarithm testing results

Generator	Results of LIL test
Standard C LCG	Fail
MT19937	Pass
PHP LCG	Fail
PHP MT19937	Fail
Flawed Debian open SSL	Fail
Standard open SSL	Pass

LIL: Law of the iterated logarithm (Wang and Nicol, 2015)

chaotic logistic maps and carefully arrange them in the same algorithm in order to increase the security level. It generates a block of 32 random bits per iteration using the following three logistic maps:

$$X_{n+1} = 3.9999\ X_n\ (1\text{-}X_n) \quad \forall n \geq 0 \tag{7}$$

$$Y_{n+1} = 3.9999\ Y_n\ (1\text{-}Y_n) \quad \forall n \geq 0 \tag{8}$$

$$Z_{n+1} = 3.9999\ Z_n\ (1\text{-}Z_n) \quad \forall n \geq 0 \tag{9}$$

The evaluation pattern is as follows:

Step 1: The speed performance analysis is achieved on a personal computer with Intel(R) Core(TM) 2 Duo CPU P7350 at 2:00 GHz. The algorithm is implemented using GCC on Fedora release 16 (Verne). Speed of this generator is 44.1120 Mbit/s and indeed the generator could pass this level successfully.

Step 2: It is generally accepted that a key space of size larger than 2^{128} is computationally secure against such attack. In this case, the size of the key space is around 2^{173}, which clearly allows resisting the brute force-attack.

Step 3: Output of this generator is used as random sample in the first level of NIST and no test is not rejected. Indeed the generator could pass this level successfully.

Step 4: About 15000 output sequences with close seed are generated for analyzing sensitivity. Correlation coefficient indicated that this generator is sensitive to seed and initial value and passes the forth level.

Step 5: Resistance against differential attack and Brute-force attack and guess-and-determine attack is approved, so it passes fifth step.

Step 6 and 7: Proportion of successful sequences in NIST test showed that the generator was successful in passing of sixth step. Also, the hypothesis of distribution uniformity of p values based on observed sequences is accepted. This means that the generator passes seventh step (Francois and Defour, 2013).

CONCLUSION

Evaluating the pseudo random number generators is essential which needs enough attention. There are a variety of criteria which are examined holistically in this study. Our suggested model

in three issues of statistical tests, resistance against attacks and security implementing considerations are emphasized. An extensive discussion was conducted on three statistical tests. A comprehensive classification for different types of attacks on pseudo random number generators was proposed. We examined issues related to security and implementing considerations. Some priorities for applying these models are proposed which can lead to good guideline for one who is trying to evaluate the generators. With following all steps of this pattern, while a comprehensive evaluation, the probability of accepting improper generators will be reduced.

REFERENCES

Akhshani, A., A. Akhavan, A. Mobaraki, S.C. Lim and Z. Hassan, 2014. Pseudo random number generator based on quantum chaotic map. Commun. Nonlinear Sci. Numer. Simul., 19: 101-111.

Alvarez, G. and S. Li, 2006. Some basic cryptographic requirements for chaos-based cryptosystems. Int. J. Bifurcation Chaos, 16: 2129-2151.

Babaei, M. and M. Farhadi, 2011. Introduction to secure PRNGs. Int. J. Commun. Network Syst. Sci., 4: 616-621.

Francois, M. and D. Defour, 2013. A pseudo-random bit generator using three chaotic logistic maps. HAL Id: hal-00785380. https://hal.archives-ouvertes.fr/hal-00785380/document.

Francois, M., T. Grosges, D. Barchiesi and R. Erra, 2013. A new pseudo-random number generator based on two chaotic maps. Informatica, 24: 181-197.

Gayakwad, R. and P. Ranbir, 2012. A pseudorandom number generator using chaotic image processing. J. Theoretical Phys. Cryptogr., 1: 6-12.

Goldreich, O., S. Goldwasser and S. Micali, 1986. How to construct random functions. J. ACM., 33: 792-807.

Goldreich, O., 2004. Foundations of Cryptography. Cambridge University Press, New York.

Hamdi, M., R. Rhouma and S. Belghith, 2015. A very efficient pseudo-random number generator based on chaotic maps and S-box tables. Int. J. Comput. Control Quantum Inform. Eng., 9: 481-485.

Ilyas, A., A. Vlad and A. Luca, 2013. Statistical analysis of pseudorandom binary sequences generated by using tent map. U. P. B. Sci. Bull., 75: 113-122.

James, F., 1990. A review of pseudorandom number generators. Comput. Phys. Commun., 60: 329-344.

Kang, J.S., 2005. Security frameworks for pseudorandom number generators. Inform. Center Math. Sci., 8: 1-11.

Kelsey, J., B. Schneier, D. Wagner and C. Hall, 1998. Cryptanalytic Attacks on Pseudorandom Number Generators. Springer-Verlag, London, UK., ISBN:3-540-64265-X, pp: 168-188.

Kelsey, J., B. Schneier and N. Ferguson, 1999. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. Proceedings of the Sixth Annual Workshop on Selected Areas in Cryptography, Kingston, Ontario, Canada, August 9-10, 1999, Springer, Berlin, pp: 13-33.

L'Ecuyer, P. and R. Simard, 2007. TestU01: AC library for empirical testing of random number generators. ACM Trans. Math. Software, Vol. 33. 10.1145/1268776.1268777

Marsaglia, G., 1996. DIEHARD: A battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/cdrom/linux/diehard.doc.

Orue, A.B., F. Montoya and L.H. Encinas, 2010. Trifork, a new pseudorandom number generator based on lagged fibonacci maps. J. Comput. Sci. Eng., 2: 46-51.

Patidar, V. and K.K. Sud, 2009. A novel pseudo random bit generator based on chaotic standard map and its testing. Electron. J. Theor. Phys., 6: 327-344.

Patidar, V., K.K. Sud and N.K. Pareek, 2009. A pseudo random bit generator based on chaotic logistic map and its statistical testing. J. Informatical, 33: 441-452.

Reyad, O. and Z. Kotulski, 2015. On pseudo-random number generators using elliptic curves and chaotic systems. Applied Math. Inform. Sci., 9: 31-38.

Rukhin, A., J. Soto, J. Nechvatal, M. Smid and E. Barker *et al*., 2010. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22 Revision 1a, National Institute of Standards and Technology, Gaithersburg, MD., April 2010, pp: 1-131.

Stoyanov, B. and K. Kordov, 2014. Novel zaslavsky map based pseudorandom bit generation scheme. Applied Math. Sci., 8: 8883-8887.

Walker, J., 2008. ENT: A pseudo-random number sequence test program. http://www.fourmilab.ch/random/.

Wang, Y. and T. Nicol, 2015. On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian OpenSSL. Comput. Secur., 53: 44-64.

Yang, L. and T. Xiao-Jun, 2012. A new pseudorandom number generator based on a complex number chaotic equation. Chin. Phys. B, Vol. 21.