Random number sequences assessment for image encryption

Antonios S. Andreatos and Apostolos P. Leros

Abstract — The aim of this study is first to present a set of tests for assessing the quality of a set of random and pseudorandom number sequences and second, to examine their suitability for image encryption. Seven different generators are considered: the pseudorandom generators of three programming languages, a chaotic random number generator, as well as, three truly random number generators. One of the truly random number generators produces poor-quality results and is used for demonstration purposes. The random number sequences were used to encrypt a test image via the bitwise XOR function. The assessment criteria used are visual tests, entropy measurements, statistical tests, image auto-correlation and cross-correlation calculations and histogram analysis. Results indicate that all but the poor-quality generators examined provide satisfactory performance.

Keywords— Image encryption, random number generator, chaotic number generator, statistical tests, correlation, histogram.

I. Introduction

The evolution of portable devices carrying digital cameras along with the availability of Internet connection, as well as, the broad use of social networks, make the production and the communication of images an everyday practice. Cyber-crime incidents have raised the issue of confidentiality; hence, image encryption has gained significant importance today. Several image encryption techniques have been proposed recently. A common yet effective simple encryption practice is to apply the bitwise Exclusive OR (XOR) between the image pixels (represented as integers in the range [0, 255]) and random or pseudorandom numbers, also in [0, 255] [1], [2]. The decryption process is just the XOR between the cipher image and the same random or pseudo random sequence.

It is the aim of this paper to compare various random and pseudo random number generators for use in image encryption, using a proposed set of tests.

A. True RNGs vs. pseudo-RNGs

A common classification of random number generators based on the source of randomness is the following [1]:

- True Random Number Generators (TRNGs),
- Pseudo-Random Number Generators (PRNGs) and
- Hybrid Random Number Generators (HRNGs).

TRNGs take advantage of unpredictable, nondeterministic sources such as natural processes or physical phenomena which can affect a sensor measuring some physical magnitude and converting the measurement into a sequence of statistically independent data. Physical phenomena commonly exploited in the generation of random numbers are radioactive decay, thermal noise and cosmic microwave background. However, the respective devices are not portable, hence unsuitable for use outside a laboratory. Therefore, good quality random numbers are often obtained by artificial sources such as the rotation of the hard disk in a computer [3], (im)properly connected diodes and transistors [4], noise collected by microphones [5], noise produced by analog radios [6] and TV sets [7], etc.

Truly random numbers are unpredictable, in the sense that it is impossible to predict the next number, given the previous numbers. For this reason TRNGs are particularly useful in cryptography and especially in key production.

PRNGs are algorithmic generators of numbers which have the appearance of randomness, but nevertheless, their results are predictable. Good random number generators produce very long sequences which look random, in the sense that no efficient algorithm can guess the next number given any prefix of the sequence. Usually, PRNGs use minimal randomness - a randomly chosen initial value called "seed". For a specific seed, PRNGs produce a specific, repeatable as well as periodic pattern [8]. This feature is desirable in cryptographic and steganographic telecommunication systems because the pseudorandom sequence used in the transmitter for encryption/ steganography must be faithfully reproduced in the receiver [2].

Both true random number generators and pseudorandom number generators, have their advantages and disadvantages. Generally, the limitation of one type is the merit of the other. The advantages and disadvantages of TRNGs are listed in Table 1. The table applies to true RNGs that are deemed to be completely random [9].

TABLE 1. ADVANTAGES & DISADVANTAGES OF TRNGS

Advantages	Disadvantages

[1]A. S. Andreatos is with the Div. of Computer Engineering & Information Science, Hellenic Air Force Academy, Dekeleia Air Force Base, Dekeleia, Attica, TGA-1010, GREECE (phone: +30-210-819-2360; e-mail: aandreatos. hafa@haf.gr, aandreatos@gmail.com).

A. P. Leros is with the Department of Automation, School of Technological Applications, Technological Educational Institute of Sterea Hellas, 34400 Psachna, Evia, GREECE (e-mail: lerosapostolos@gmail.com).

Non periodical	Slow and inefficient
No predictability of random numbers based on knowledge of preceding sequences	Cumbersome to install and use
Certainty that there are no dependencies present	Random number sequences are not reproducible
High level of security	Relatively costly
Based on a natural process – not based on algorithm	Possibility of manipulation

The disadvantages of the TRNGs constitute advantages of PRNGs (and vice-versa) are listed in Table 2:

TABLE 2. ADVANTAGES & DISADVANTAGES OF PRNGS

Advantages	Disadvantages
Fast	Output is periodical
Easy to install and use	Output is predictable
Reproducible output	Output depends on seed
Low Cost	Low level of security
Highly portable, easy to change	Based on some algorithm

In practice good TRNGs are hard to find, hard to proof, implemented in hardware, hence often expensive, and, most important for telecom applications, non-reproducible (in the receiver). Therefore, in most applications such as decision making, software testing, simulation and cryptography, PRNGs dominate.

A special case of TRNGs takes advantage of the (computer) user behaviour to generate random numbers; the system generates values out of keyboard inputs, mouse movements, and/or other human actions. Linux provides /dev/random for this purpose. A very simple but effective way to obtain a seed is getting the time from the system's Real Time Clock, or a function of it [10]. In fact, one of the TRNGs implemented in this research employs exactly this methodology.

Hybrid random number generators combine methods of TRNGs and PRNGs to calculate random numbers. For instance, they may use a true random number as an initial seed for an algorithm that generates pseudorandom numbers. This is the most commonly used method in Operating Systems because it is fast and flexible [10]. Linux provides /dev/urandom for this purpose.

A special category of generators contains the chaotic random number generators (CRNGs) which are based on chaotic phenomena [1]. Physical implementations of chaotic generators (such as those based on electronic circuits) approach TRNGs because real device values have a tolerance and they are also affected by environmental reasons, aging, etc. Software simulations of chaotic phenomena resemble PRNGs, hence they share the same advantages and can be used in cryptography [2] and steganography [11]. The most famous as well as simple chaotic implementation is Chua's circuit [12]. Several cryptographic and steganographic telecommunication systems based on Chua's circuit and its variations have been proposed [1], [2], [11], [12]. Various methods for producing good quality random number sequences (abbreviated as RNS henceforth) based on Chua's circuit have been proposed [1], [14], [15].

In this paper we compare seven sources of random numbers and their suitability in image encryption. For demonstration purposes, the first source produces poor quality RNS.

1/ A poor quality RNS obtained from a physical source without post-processing.
2/ The PRNG of C (rand() function).
3/ The PRNG of PHP (rand() function).
4/ The PRNG of Matlab (randi function).
5/ A chaotic PRNG based on Chua's circuit, simulated in Matlab, abbreviated henceforth as CRNG [15].
6/ A truly random generator called “haveged”, based on the HAVEGE algorithm [16], [17], [18] (see also Haveged manual pages).
7/ A truly random RNS obtained from recorded digitised AM radio noise, after proper post-processing [6].

The HAVEGE (HArdware Volatile Entropy Gathering and Expansion) algorithm harvests the indirect effects of hardware events on hidden processor state (caches, branch predictors, memory translation tables, etc.) to generate a random sequence. The effects of interrupt service on processor state are perceived as timing variations in program execution speed. Using a branch-rich calculation that fills the processor instruction and data cache, a high resolution timer source such as the processor time stamp counter can generate a random sequence even on an idle system.

The number sequences produced by the aforementioned sources 2 to 7 are uniformly distributed [19], i.e., they have equal probability of appearance, a fundamental property of true random numbers.

The quality of random number sequences (RNSs) is critical in many security applications including cryptography and steganography. Therefore, many kinds of tests have been devised and are used to measure the quality of RN sequences, hence, the corresponding generators. Some of the most common tests are Entropy tests, Statistical Tests, etc. Some of the most common test suites are: the FIPS 140-2 [20], the NIST suite [21], the AIS-31 [22], TestU01 [23], etc.

The tests considered in this paper are the following:
1/ Visual Tests checking distribution;
2/ Entropy tests;
3/ Statistical tests;
4/ Image correlation tests;

5/ Histogram analysis.

The FIPS 140-2 Test suite was used for the Statistical Tests. The Federal Information Processing Standard (FIPS) Publication 140-2 (FIPS PUB 140-2) is a U.S. government computer security standard used to accredit cryptographic modules [20]. The official title is Security Requirements for Cryptographic Modules. Initial publication was on May 25, 2001 and was last updated December 3, 2002.

This standard provides increasing, qualitative levels of security intended to cover a wide range of potential applications and environments. The security requirements cover areas related to the secure design and implementation of a cryptographic module.

This research was implemented on Linux platform.

B. Test image

The test image is a colour jpeg image with dimensions 153x348x3 (colours) = 159732 bytes. It is shown in Fig. 1.

Fig. 1 Test image

A good encryption scheme should produce a cipher image that looks like rubbish, i.e. random bytes which produce a result like noise. Figure 2 presents a poorly encrypted image which fails to hide the original information.

Fig. 2 Poorly encrypted image

In order for these random sequences to be used for cryptography or steganography, it is desirable to be identically reproduced in the receiver. This can be easily achieved for algorithmic generators such as the PRNGs.

So in this paper we present a framework for assessing the quality of image encryption; we also use this framework for comparing a set of random and pseudo-random sequences for image encryption.

The tests considered here are the following: section II deals with the autocorrelation of the visual tests; section III deals with entropy tests; section IV deals with statistical tests; section V deals with image correlation tests; section VI deals with the assessment of the encrypted image (histogram analysis). We conclude in section VII. This is an evolution of our previous work [19] and fulfills most of the future research specified therein.

II. Visual Tests

Visual tests constitute an easy and quick way for humans to check the characteristics of an image and the represented parameter or magnitude. In [19] we have examined two visual tests: a) Uniformity test of the produced RNSs and b) Cipher-image inspection. All generators examined there produced satisfactory results.

Our generators should not be biased, i.e., they should produce random numbers with equal probability, or else, all possible numbers [0 - 255] should have the same frequency of appearance. Figures 3-9 of [19] demonstrate the histograms of the number sequences under test, obtained by means of the Binary Viewer software. All generators produce satisfactory results. Here we propose two additional visual tests.

A. Scatter diagrams tests

Scatter diagrams allow us to visualize the distribution of the produced random numbers in space [8]. This allows us to check the uniformity of the produced RNS. Under certain conditions we may observe artifacts [19] or even periodicity [24].

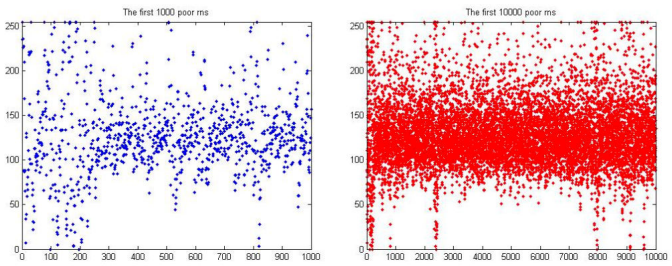

Fig. 3 Scatter diagrams of poor quality random numbers

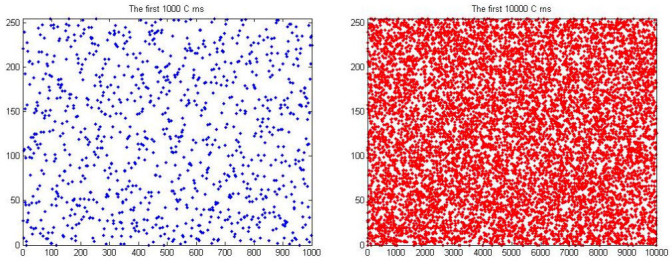

Fig. 4 Scatter diagrams of C random numbers

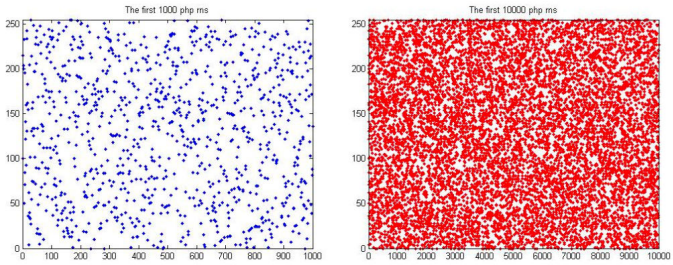

Fig. 5 Scatter diagrams of PHP random numbers

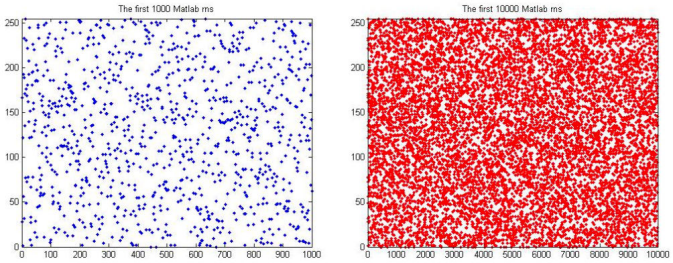

Fig. 6 Scatter diagrams of Matlab random numbers

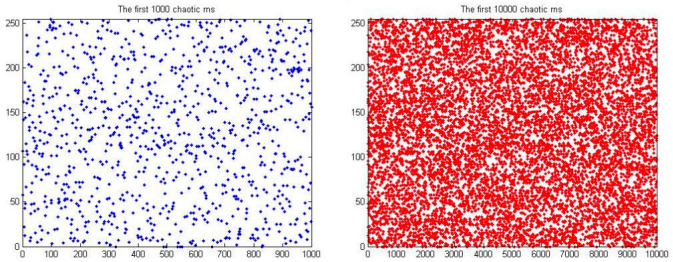

Fig. 7 Scatter diagrams of CRNG random numbers

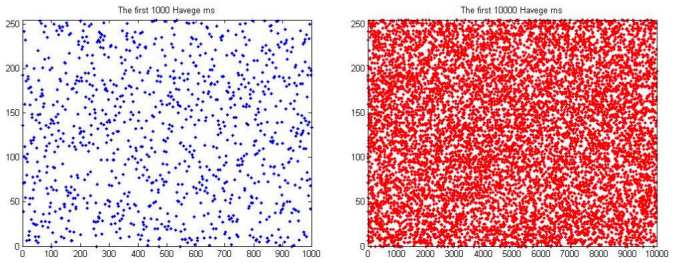

Fig. 8 Scatter diagrams of Havege random numbers

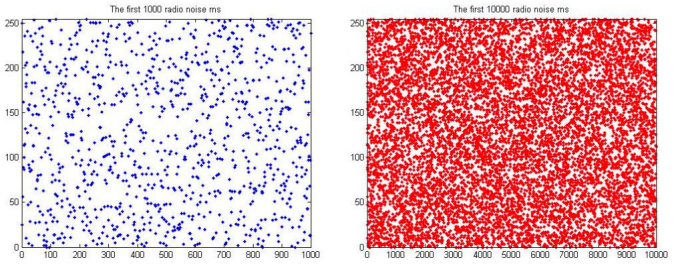

Fig. 9 Scatter diagrams of Radio noise random numbers

Comparing the images we can conclude that all sequences are uniform, but the CRNG produces a coarser histogram. This is also verified by the difference Max Frequency - Min Frequency in the above figures. CRNG random number sequence has the largest difference.

B. Cipher-image inspection

In this test (Figures 10-16) we observe the cipher-images produced by the bitwise XOR between the original (test) image and the random number sequences [19].

Fig. 10 Cipher-image produced by poor quality RNS

Attentive readers will discern the figure of the airplane in this cipher image; moreover, the noise is coarser compared to the following cipher images.

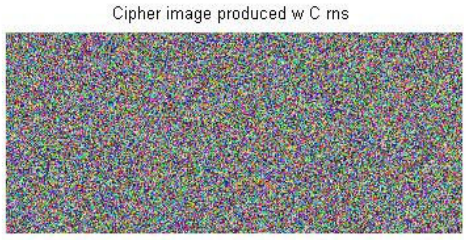

Fig. 11 Cipher-image produced by C RNS

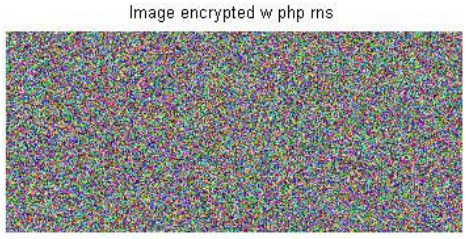

Fig. 12 Cipher-image produced by PHP RNS

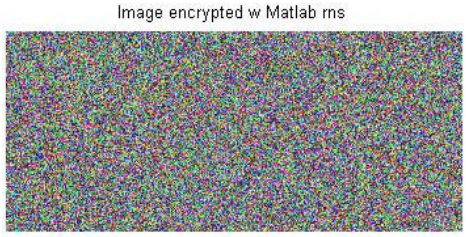

Fig. 13 Cipher-image produced by Matlab RNS

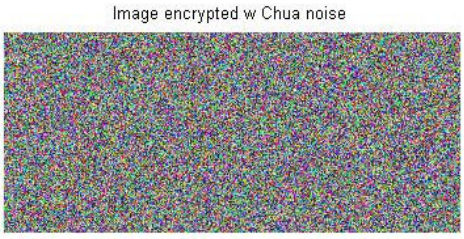

Fig. 14 Cipher-image produced by CRNG RNS

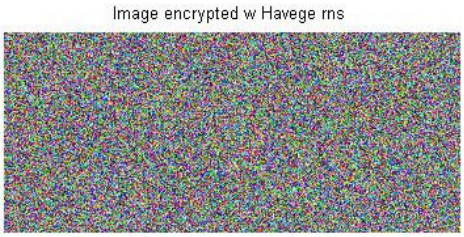

Fig. 15 Cipher-images produced by Havege RNS

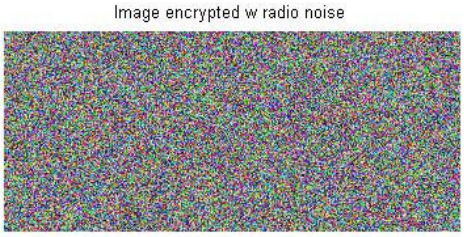

Fig. 16 Cipher-images produced by Radio noise RNS

From the presentation of the cipher-images we may conclude that all but the poor quality RNSs succeed to hide the information, since they all look like random noise.

III. Entropy tests

In information theory, entropy is defined as the amount of information contained in a random variable [5].

Shannon denoted the entropy H of a discrete random variable X with possible values $\{x_1, \ldots, x_n\}$ as H(X) = E(I(X)). Here E is the expected value, and I is the information content of X. I(X) itself is a random variable. If p denotes the probability mass function of X then the entropy can explicitly be written as [25]:

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i)$$

$$= \sum_{i=1}^{n} p(x_i) \log_b \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \quad (1)$$

Entropy tests were performed by means of the ***ent*** bash command, as well as, the Binary Viewer software. The higher the Entropy, the better the quality of random numbers. Results are as shown in Table 3 below in bits per byte. Entropy of a truly random bit sequence equals its size in bits; hence, a result of 8 bits per byte means perfect. In Binary Viewer (BV) the perfect entropy is represented by 1.

TABLE 3. ENTROPY TESTS

	Poor Qual RNS	C	PHP	Matlab	CRNG	HAVEGE	Radio noise
Entropy (BV) (best = 1)	0.457327	0.999850	0.99987	0.999849	0.9988874	0.999864	0.999861
Entropy (ent) (best = 8)	6.942329	7.998804	7.998959	7.998791	7.9990997	7.998738	7.998886
Chi square distribution	255133	265.04	230.47	268.08	2053.5	279.93	247.21
Arithmetic mean	127.8622	127.2457	127.5162	127.4706	127.5267	127.6865	127.5713
Monte Carlo value for Pi[2]	3.87754488 8	3.15273082 4	3.159942 9	3.14025993 5	3.13214634 5	3.133782 77	3.142062955
error %	23.43	0.35	0.58	0.04	0.30	0.25	0.01
Serial correlation coefficient	0.646444	-0.003116	-0.002326	-0.000464	-0.000630	0.002261	0.0012

IV. Statistical Tests

In this work we deal with the most important statistical tests. Table 4 briefly explains each test. For a more detailed description and a step by step guide, the reader is referred to Appendix K of [9].

TABLE 4. EXPLANATION OF COMMON TESTS

Test	Defect detected
Monobit	Too many zeroes or ones (over 50 ± 0.0346%)
Poker[3]	Check for certain sequences of five numbers
Runs[4]	Oscillation of zeroes and ones is too fast or too slow (locally)
Long runs	Oscillation of zeroes and ones is too fast or too slow (globally)

[2] http://en.wikipedia.org/wiki/Monte_Carlo_method#Monte_Carlo_and_random_numbers.

[3] http://en.wikipedia.org/wiki/Statistical_randomness.

[4] See http://www.cs.indiana.edu/~kapadia/project2/node17.html.

The results of statistical tests, obtained by means of the *rngtest* bash shell script[5], are presented in Table 5. Surprisingly enough, the C data set passes all tests, surpassing all other generators.

TABLE 5. STATISTICAL TESTS

	Poor Qual RNS	C	PHP	Matlab	CRNG	HAVEGE	Radio noise
Successes	0	63	63	58	59	63	63
Failures	63	0	0	5	4	0	0
Monobit	21	0	0	0	0	0	0
Poker	63	0	0	0	1	0	0
Runs	63	0	0	0	0	0	0
Long run	0	0	2	5	3	0	0
Continuous run	0	0	0	0	0	0	0

We observe here that Matlab RNS and CRNS fail in some tests.

V. IMAGE CORRELATION TESTS

A. Auto-correlation Analysis

In this section we compare the autocorrelation of the encrypted images. The autocorrelation of an image is defined here as the similarity of an image with itself, shifted by one pixel horizontally, vertically, diagonally and anti-diagonally. The autocorrelation was calculated in Matlab using the following formulae (2) and (3) [2]. In general the correlation coefficient γ for a pair of pixels is defined as described in formula (2):

$$\gamma(x,y)=\frac{\operatorname{cov}(x,y)}{\sigma_x\sigma_y} \tag{2}$$

Where:

$$\operatorname{cov}(x,y)=\frac{1}{N}\sum_{i=1}^{N}\left[x_i-E(x)\right]\left[y_i-E(y)\right] \tag{3}$$

In eq. (3) "E" is the expected value operator and in eq.2 σ_x^2 represents the variance of variable x. The values of γ(x,y) lie in the range [−1, 1], with 1 indicating perfect correlation, −1 indicating perfect anti-correlation and 0 indicating no correlation. The evaluation of these formulae in MATLAB was realised by the use of built-in functions.

TABLE 6. AUTO-CORRELATION TESTS

	Horizontal	Vertical	Diagonal	Anti-diagonal
Original image	0.9882	0.9580	0.9485	0.9565
Poor qual. RNS	-0.0091	0.2871	0.0037	-0.0050
C	0.0007	-0.0000	-0.0061	-0.0050
PHP	0.0025	0.0008	0.0000	0.0033
Matlab	0.0001	0.0002	0.0051	-0.0029
CRNG	-0.0017	-0.0039	0.0016	-0.0009
Havege	0.0019	-0.0000	0.0014	0.0000
Radio noise	-0.0007	-0.0003	-0.0012	-0.0024

As expected, the autocorrelation of the original image is close to 1. This practically means that image pixels (hence image rows and columns) have a great probability to be similar to adjacent pixels (or rows or columns).

However, the autocorrelation of all the encrypted images is close to 0, fact that implies that they do not represent something meaningful but instead they look like nonsense or rubbish. This is in accordance to the visual tests of the cipher-images. Hence, all generators produce satisfactory results.

B. Cross-correlation Analysis

Another test that we can perform is to check the similarity of each cipher-image with the original image. Mathematically this is expressed by the cross-correlation coefficient [26]. The cross-correlation coefficients were computed by means of MATLAB built-in functions. The results obtained are presented in Table 7, where the original image has been included for comparison purposes.

TABLE 7. CROSS-CORRELATION TESTS

RNG	Cross-correlation coefficient
Original image	1.0000
Poor quality RNS	0.0530
C	0.0030
PHP	0.0005
Matlab	-0.0002
CRNG	-0.0018
Havege	0.0026
Radio noise	0.0042

As we can see from Table 7, the original image is 100% correlated with itself whereas the poorly encrypted image is 5.3% correlated with the original image. The rest cross-correlation coefficients are close to zero, hence indicate good encryption since 0 indicates no correlation.

[5] See http://www.linuxcommand.org/man_pages/rngtest1.html.

VI. Histogram Analysis

An important characteristic of an image is its histogram. The histogram uniquely characterises the corresponding image, so a robust encryption scheme should produce a completely different histogram. An ideal image encryption scheme should produce an output image totally uncorrelated from the original image with a histogram totally independent from the input image. In other words, the histogram of the output image should be invariant, independent from the input image, and a flat histogram satisfies these requirements.

Figure 17 presents the histogram of the original (Red colour only) image.

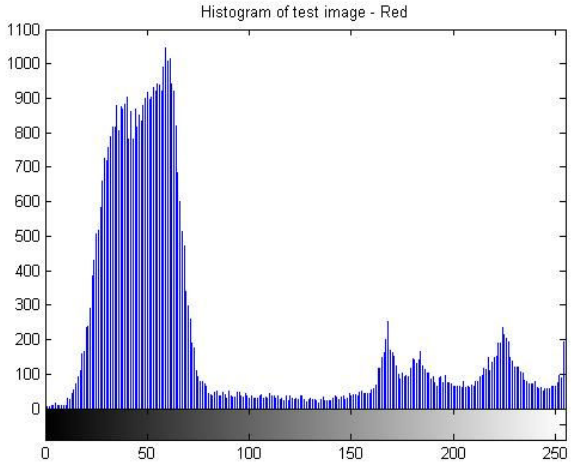

Fig. 17 Histogram of the original image

In this section the histograms of the cipher-images will be analysed. Since the images used here are colour, there are three distinct histograms for each image, one for each colour component (R, G, B). For brevity purposes, only the red histograms will be presented; however, similar results are obtained for the green and blue histograms as well.

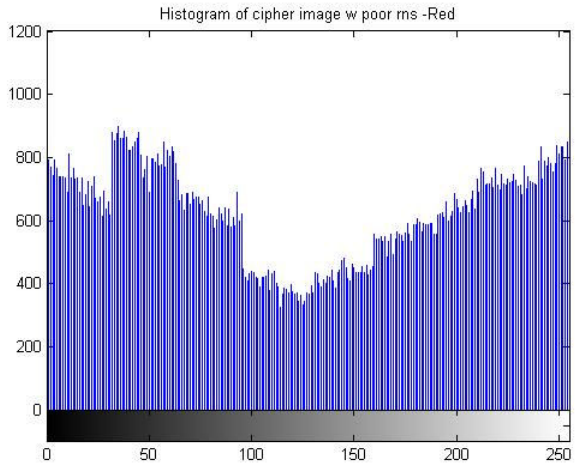

Fig. 18 Histogram of image encrypted by poor quality RNS

Figure 18 presents the histogram of the poorly encrypted image. Observe that this histogram is not flat.

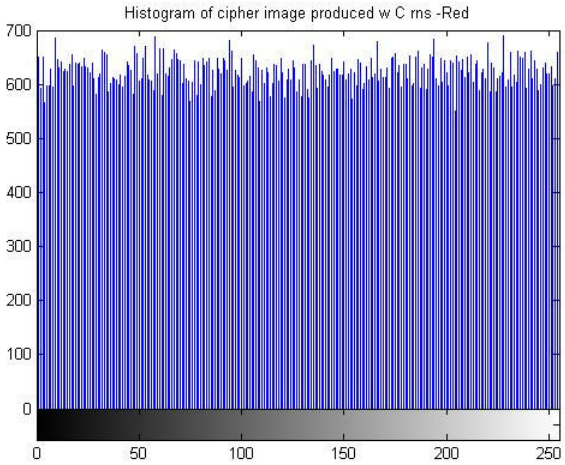

Fig. 19 Histogram of image encrypted by C RNS

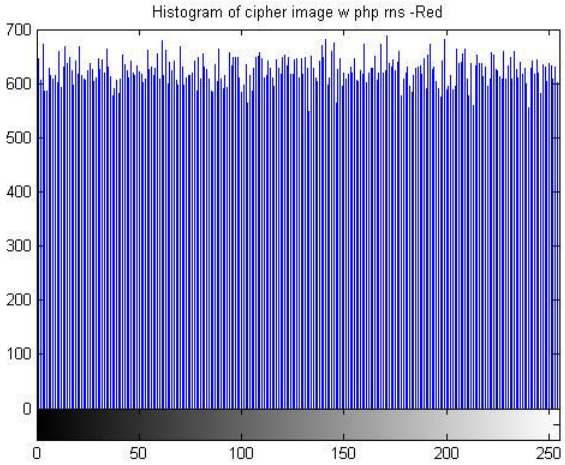

Fig. 20 Histogram of image encrypted by PHP RNS

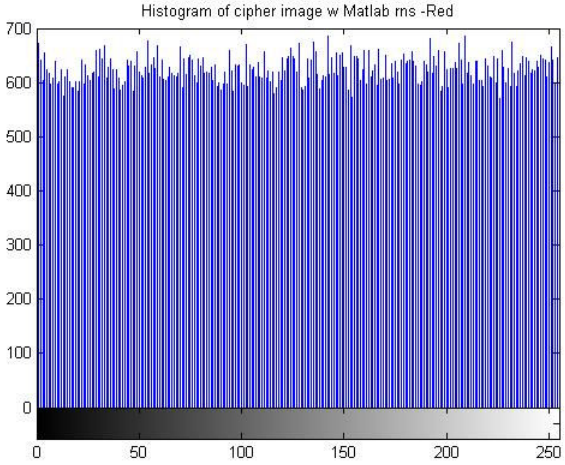

Fig. 21 Histogram of image encrypted by Matlab RNS

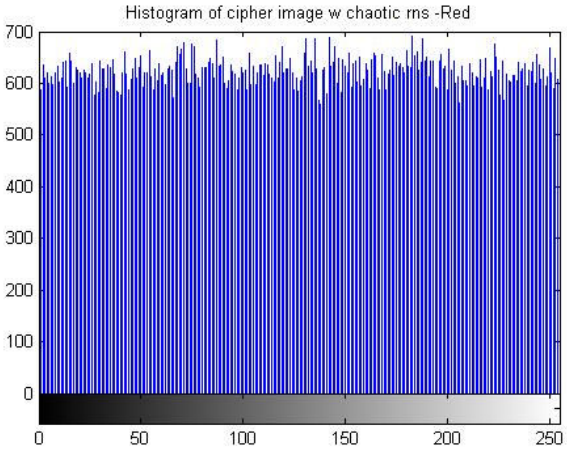

Fig. 22 Histogram of image encrypted by CRNG

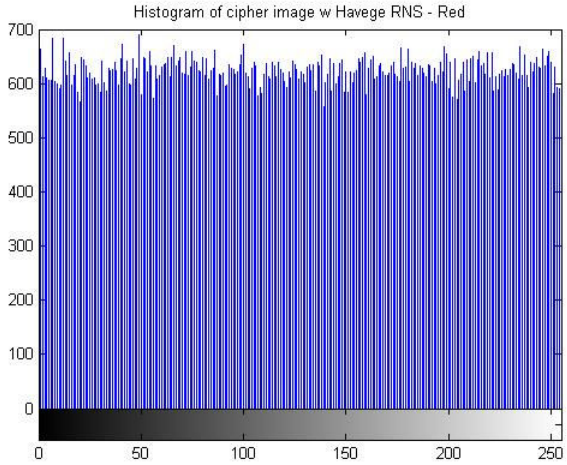

Fig. 23 Histogram of image encrypted by Havege RNS

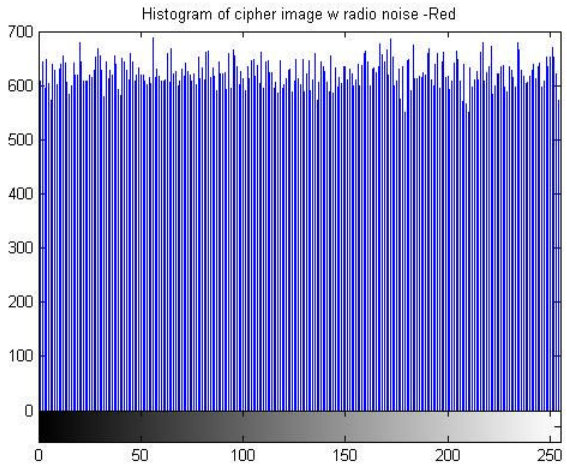

Fig. 24 Histogram of image encrypted by Radio noise

From the presentation of the various histograms we can see that all RNSs satisfy the requirement for a flat distribution regardless the input image, hence they produce good cipher-images.

VII. Conclusion

In this paper (and partially in [19]) we have presented a series of tests for assessing random number sequences for image encryption. These tests are:

1/ Visual tests;
- 1a) Uniformity tests (see [19]);
- 1b) Cipher-image inspection tests (see [19]);
- 1b) Scatter diagrams tests;
- 1c) Cipher-image inspection;

2/ Entropy tests;
3/ Statistical tests;
4/ Image correlation tests;
- 4a) Auto-correlation tests (see [19]);
- 4b) Cross-correlation tests;

5/ Histogram analysis.

The specific set of tests comprises a universal and multi-dimensional framework for evaluating image encryption algorithms. Using this test set we have assessed a series of random number sequences produced by three pseudo-random number generators, two truly random number generators set, plus a chaotic RNG. We have also included a poor-quality RNS for demonstration purposes. From the comparison it is concluded that all sources except the poor-quality one, produce acceptable, high quality results, for the selected test image.

All generators produce satisfactory and comparable results, which means that the specific generators are of good quality (except the poor-quality one). In fact, PHP and Matlab use the Mersenne twister generator which is, by far, the best and most widely used PRNG [27]. Mersenne twister is used by many programming languages, including PHP and Matlab; hence the good results. The most commonly used version of Mersenne Twister is "MT19937", which has a very long period of $2^{19937}-1 \approx 4.3154248\times10^{6001}$.

As stated in the Introduction, true random number sequences are not reproducible; hence, it will be impossible to reproduce a given RNS used at the transmitting side for encryption, to decode the cipher-image at the receiver.

On the other hand, nor pseudo-random numbers are suitable for cryptography, for a number of reasons. Pseudo-random numbers are in fact deterministic, periodic sequences [8], [24], hence, easier to guess/ break. Moreover, they are subject to a number of potential problems such as shorter periods for some seed values, correlation of successive values, etc. [28].

Observing a sufficient number of past iterations allows us to predict all future iterations [27]. Hence, they are less suitable for cryptography. Another disadvantage is that their output is a function of a single parameter, i.e., the seed [9].

The CRNG produces satisfactory results, comparable with the PRNGs and the TRNG; moreover, it has some additional

advantages:

- They are periodic but by applying spatiotemporal techniques, their period may become some years long, hence practically unreachable [29];
- Under proper design of the generator, they can use multi-parametric keys rather than a single seed (see for instance [2], [11];
- The CRNG is multi-parametric system and must be fine-tuned in order to improve its performance. A method for tuning Chua-based CRNGs has been proposed in [30].

In conclusion, in this paper we have demonstrated a simple yet effective method for image encryption, based on the bitwise XOR function. The proposed method is powerful, fast and easy to implement, and may employ a variety of random as well as pseudo-random generators. We have also proposed a framework for testing the RNSs and the cipher-images. Using this framework we have verified that all generators under test (except the poor-quality one) produce satisfactory cipher-images from a security viewpoint.

REFERENCES

[1] C. K. Volos, "Image Encryption scheme based on coupled Chaotic systems", in *JAMB,* 3, 1, 2013, pp. 123-149.

[2] A. Andreatos and A. Leros, "Secure image encryption based on a Chua chaotic noise generator. Journal of Engineering Science and Technology Review". *JESTR Special Issue on Nonlinear Circuits: Theory and Applications,* 6 (4) (2013) 90-103. Available: http://jestr.org/index.php?option=com_content&view=article&id=31&Itemid=71.

[3] D. Davis, R. Ihaka and P. Fenstermacher, "Cryptographic randomness from air turbulence in disk drives", Proc. CRYPTO' 94, pp. 114 - 120. Available: theworld.com/~dtd/random/forward.ps.

[4] Leon's Mini Random Number Generator [Online]. Available: http://www.physics.wisc.edu/~lmaurer/projects/minirng/minirng.html

[5] S. Theodoulou, "*Improving the Reliability of Cryptographic Services in Personal Computers using Truly Random Numbers and Advanced Coding Techniques*", Diploma Thesis, Hellenic Air Force Academy, June 2010 (in Greek).

[6] "It did not succeed, it just happened", *Delta Haker magazine*, no. 33, pp. 68-82, June 2014 (in Greek).

[7] "Randomness generators", *Delta Haker magazine*, no. 31, pp. 62-68, Apr. 2014 (in Greek).

[8] A.N. Veneti, G.C. Meletiou and M.N. Vrahatis, "Fractal Dimension As An Assessment Metric for Pseudorandom Number Generators". Presented at the *2nd International Conference on Cryptography, Network Security and Applications in the Armed Forces*. Hellenic Military Academy, April 2, 2014.

[9] Charmaine Kenny, "Random Number Generators: An evaluation and comparison of Random.org and some commonly used generators". Project Report, supervised by Krzysztof Mosurski. Trinity College Dublin, April 2005.

[10] http://wiki.osdev.org/Random_Number_Generator.

[11] A. P. Leros and A. S. Andreatos, "A Video Steganography System for Secure Data Communication", Chapter 4.3 in *New Research Trends in Nonlinear Circuits: Design, Chaotic Phenomena and Applications*. Nova Science Publishers, Inc. 2014.

[12] L. O. Chua, "Chua's circuit: ten years later", *IEICE Trans. Fundamentals* E77-A, 11, pp. 1811-1822 (1994).

[13] L. Gámez-Guzmán, C. Cruz-Hernández, R.M. López-Gutiérrez, E.E. García-Guerrero, Synchronization of Chua's circuits with multi-scroll attractors: Application to communication, *Commun. Nonlinear Sci. Numer. Simulat*. 14, 2765–2775 (2009).

[14] M. E. Yalçın, J. A. K. Suykens and J.Vandewalle, "True Random Bit Generation From a Double-Scroll Attractor", *IEEE Transactions on Circuits and Systems* —I: Regular Papers, 51, 7, pp. 1395-1404 (2004).

[15] A. S. Andreatos and C. K. Volos, "Secure Text Encryption Based on Hardware Chaotic Noise Generator". Presented at the *2nd International Conference on Cryptography, Network Security and Applications in the* Armed Forces, Hellenic Military Academy, April 2, 2014.

[16] A. Seznec and N. Sendrier, "HArdware Volatile Entropy Gathering and Expansion: generating unpredictable random numbers at user level", INRIA Research Report, RR-4592, October 2002.

[17] Haveged - A simple entropy daemon [Online]. Available: http://www.issihosts.com/haveged/index.html.

[18] http://www.issihosts.com/haveged/faq.html.

[19] A. S. Andreatos and A. P. Leros, "A comparison of random number sequences for image encryption", in *Proceedings of MMCTSE, Mathematical Methods & Computational Techniques in Science & Engineering*. Athens, Greece, November 28-30, 2014, pp. 146-151.

[20] *Security Requirements For Cryptographic Modules*, 2001 [Online]. Available: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

[21] NIST Special Publication 800-22, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". Available: http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf.

[22] AIS 31, "Functionality Classes and Evaluation Methodology for Physical Random Number Generators", Version 1 (25.09.2001) [Online]. Available: www.bsi.bund.de/zertifiz/zert/interpr/ais31e.pdf.

[23] TestU01, [Online]. Available: http://www.iro.umontreal.ca/~simardr/testu01/tu01.html.

[24] A. Divyanjali and V. Pareek, "An Overview of Cryptographically Secure Pseudorandom Number generators and BBS", *International Journal of Computer Applications* (IJCA) (0975 – 8887), pp. 19-28, 2014.

[25] T. M. Cover and J. A. Thomas, "Elements of Information Theory", 2nd edition. Wiley, 2006.

[26] A. Belmeguenaï, K. Mansouri and L. Grouche, "Implementation of Blum Blum Shub Generator for Message Encryption", in *Proceedings of International Conference on Control, Engineering & Information Technology* (CEIT'14), pp. 118-123.

[27] Mersenne twister, http://en.wikipedia.org/wiki/Mersenne_twister.

[28] http://en.wikipedia.org/wiki/Pseudorandom_number_generator#Potential_problems_with_deterministic_generators.

[29] S. Wang, W. Liu, H. Lu, J. Kuang and G. Hu, "Periodicity of chaotic trajectories in realizations of finite computer precisions and its implication in chaos communications", *International Journal of Modern Physics B*, Vol. 18, Issue 17, no. 19, 30 July 2004, pp. 2617-2622.

[30] A. Leros and A. Andreatos, "On the optimisation of Chua chaotic attractors", in *Proceedings of ICACM '13, 2nd WSEAS International Conference on Applied and Computational Mathematics*, Vouliagmeni, Athens, Greece, May 14-16, 2013. Available: http://www.wseas.org/wseas/cms.action?id=2574.