International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 6, December 2019, pp. 5615~5627
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i6.pp5615-5627

Cryptographic adaptation of the middle square generator

Hana Ali-Pacha[1], Naima Hadj-Said[2], Adda Ali-Pacha[3], Mohamad Afendee Mohamed[4], Mustafa Mamat[5]
[1,2,3]Lab. of Coding and Security of Information (LACOSI), University of Sciences and Technology of Oran, Algeria
[4,5]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut Campus, Malaysia

Article Info

Article history:

Received Apr 5, 2019
Revised Jul 19, 2019
Accepted Jul 28, 2019

Keywords:

Cryptography
Entropy test
Middle square generator
Random nicknames
Von neumann

ABSTRACT

Currently, cryptography plays a major role in various computer and technological applications. With the high number of internet users, the use of cryptography to provide information security has become a priority. Several applications such as e-mails, electronic banking, medical databases and e-commerce require the exchange of private information. While, if the connection is not secure, this sensitive information can be attacked. The best-known cryptographic systems rely on the generation of random numbers, which are fundamental in various cryptographic applications such as key generation and data encryption. In what follows, we want to use pseudo-random sequences generated by the middle square generator. In this work, it must be possible to estimate that the data produced has random characteristics, knowing that the algorithm used is deterministic. Overall, this paper focuses on the testing of pseudo-random sequences generated by the middle square generator and its use in data encryption.

Corresponding Author:

Hana Ali-Pacha,
Lab. of Coding and Security of Information (LACOSI),
University of Sciences and Technology of Oran,
Po Box 1505 Oran M'Naouer Algeria.
Email: hana.alipacha@univ-usto.dz

1. INTRODUCTION

The development of algorithms generating pseudo-random numbers is very much related to that of cryptography [1-8]. Especially, the militarily importance such as communication and monitoring [9-10] of this science have motivated many researches throughout history. But there is no pseudo-random algorithm that can escape from statistical analysis, especially because the "seed" must theoretically itself be random, and the algorithm used cannot be initialized by itself. The current cryptographic generators are thus obliged to include element that is not generated in a deterministic way. One thus moves towards hybrid generators, founding a robust algorithm of pseudorandom number generation by initializing itself through a physical means of chance production.

On the other hand, the image becomes more and more indispensable in several fields and essentially in communication between people. Indeed, the exponential development of communication media on the one hand, and digital storage media on the other hand, have enormously transformed the way we communicate. These new technologies are based essentially on the efficient exchange and storage of multimedia data and in particular digital images, hence the need for image encryption algorithms.

In what follows, we discuss the design and realization of the middle-square generator in the context of producing pseudo-random sequences. The simplistic of middle-square generator can be exploited for good random sequence. In order to evaluate these sequences and validate our generator, we implemented five static tests.

This paper is structured as follows. Section 2 introduces some basic principles of pseudo-random number generator. Section 3 discusses the formation of middle square generator, outlining the procedure with an examples. Section 4 presents various types of testings purposedly to evaluate the quality of our proposed method. Section 5 reports the findings according to various tests proposed earlier Section 6 concludes our studies.

2. PSEUDO-RANDOM GENERATORS

The need for random numbers is felt in many applications of cryptography. In common cryptographic systems, the keys (numbers) that are used must be randomly generated. For example, when one consults on the Internet his e-mail accounts or, when one carries out an order by Internet some "sensitive" information (your access code or your credit card number) must remain confidential to ensure authenticity, nobody should be able to access the accounts or order with the card. To ensure these functions, Internet protocols have been put in place. They allow you to enter your codes on a web page without the risk of an outside person having access to them. These protocols use random numbers to encrypt data and prevent spying.

2.1. Definition of a random sequence

In mathematics, a random sequence, or random infinite sequence, is a sequence of symbols of an alphabet having no structure, no regularity, or identifiable prediction rule [11-12]. Such a sequence corresponds to the intuitive notion of numbers drawn at random. A sequence of random numbers is a sequence of numbers randomly chosen. This sequence has the property that we cannot predict the numbers to come from the already known numbers, whatever they are [13].

2.2. Definition of a pseudo-random sequence

The pseudo-random term is used in mathematics and computer science to designate a sequence of numbers that approaches a statistically perfect hazard [14-16]. By the algorithmic processes used to create it and the sources used, the sequence cannot be completely considered as truly random. A pseudo-random sequence (Pseudo Random Sequence in English) [17] is a sequence of integers x_0, x_1, x_2, ... taking its values in the set M = {0, 1, 2, ..., m-1}. The term x_n (n> 0) is the result of a calculation (to be defined) on the previous term (s). The first term x_0 is called the seed. With the same initial seed, the sequence of pseudorandom numbers produced by the pseudorandom number generator is deterministic and can therefore be reproduced.

A pseudo-random number generator is an algorithm that generates a sequence of numbers with certain properties of chance. The principle of these generators is to create from an initial seed, a so-called pseudo-random number, which has no apparent logical or arithmetic connection with the seed. This generated number is then used to create a second pseudo-random number. We can thus recursively generate a series of numbers that do not appear to have any logical link in their sequence, but which are in fact all obtained by a deterministic formula. This class of generators is easy to implement and allows high throughputs while producing suites that have good statistical properties. It is therefore very suitable for applications that do not require the unpredictability of the suites (such as digital simulation), but can also be used in cryptographic applications provided that certain criteria are met.

3. MIDDLE SQUARE GENERATOR

This generator is based on the median square method, known in the English literature also as middle square, was invented by the American-Hungarian mathematician and physicist John Von Neumann in 1946. The middle square is considered as the first method of automatic generation of pseudorandom numbers. The principle of this method is very simple, we generate a sequence of numbers each having 2k digits (even number). The successor of a number in this sequence is obtained by raising this number squared and then retaining the 2k middle numbers. The principle of this method is described by the following steps:

- Start with a seed (a number) of n-digit (n digits),
- Raise squared to get a number of 2n digits, add zeros if necessary,
- Take the middle n numbers as the next random number,
- Repeat 1-2-3 (the process).

Example : If we generate 4-digit numbers from 3567, we get 7234 as the next value since the 3567 square is equal to 12723489, continuing in the same way, the next number will be 3307, 3567, 7234, 3307, 9362 , 6470, 8609, 1148, 3179, 1060, 1236, 5276, 8361, 9063, 1379, 9016, 2882, 3059, 3574, 7734, 8147, 3736, 9576, 6997, 9580, 7764, 2796, 8176, 8469, 7239 , 4031, 2489, 1951, 8064, 280, 784, 6146, 7733, 7992, 8720, 384, 1474, 1726, 9790, 8441, 2504, 2700, 2900, 4100, 8100, 6100, 2100, 4100, and so on. The resulting sequence returns after 46 iterations in a periodic orbit.

4. TEST A GENERATOR

Determining whether a generator is random or not is a tricky problem. Indeed, there is no universal test, that can say with certainty that a generator is random [18-19]. The principle is to show that it is not biased by studying the properties of the numbers it generates. In practice, a random generator produces a sequence of numbers with properties of unpredictability [20-22] and independence, and follows a certain distribution (uniform in cryptography, Gaussian in telecommunications, etc.). The evaluation of the random quality of a generator thus passes through the control of the properties of the sequence that it generates. This is achieved through statistical tests that compare the performance of the generator studied compared to those, theoretical.

The purpose of statistical tests is to measure the quality of a random sequence. We can conclude that a suite generated by a PRNG is random and of good quality, if it satisfies these tests. Therefore, a statistical test can in no way guarantee that a given sequence is random. The only information that a statistical test can provide is that the sequence seems random. Several standards exist to evaluate and certify the quality of pseudorandom number generators. We will present some tests used to evaluate the performance of our generator.

4.1. Entropy test

An entropy calculates the amount of information contained in a file. The file is considered as a sequence of words of 1 or 8 bits. The entropy is calculated as shown below:

$$H(x) = -\sum_{i=0}^{2^n-1} P_i(X).log_2(P_i\,(X))$$

Where X is the studied source, P_i is the probability of appearance of the word i of n bits. The computation of the entropy makes the minimum number of bits per word containing all the information. For example, if the entropy is 6 bits/word for 8-bit words then 2 bits carry redundant information and the file could theoretically be compressed to three quarters of its original size.

4.2. Mean, standard deviation and auto-correlation factor

This is the simplest test possible. It consists of calculating the mean, the variance and the autocorrelation factor of the pseudo-random sequence. Let x_i, for i = 1, 2, .., n, be a sequence obtained from a pseudo-random number generator. The sequence of $u_i=x_i/n$, for i = 1, 2, .., n is a sequence of pseudo-random numbers distributed uniformly in the interval [0,1]. For a random sequence, these factors tend towards ideal values, which are thus sufficient to compare with the calculated values for the following U. Ideally, we must find the three values below:

a) Average: $\mu = \frac{1}{n}.\sum_{i=1}^{n} u_i = \frac{1}{2}$.

b) Variance: $\vartheta = \frac{1}{n}.\sum_{i}^{n}({u_i}^2 - \mu^2) = \frac{1}{12}$.

c) Auto Correlation: $r = \frac{1}{n}.\sum_{i}^{n-1}(u_i * u_{i+1}) = \frac{1}{4}$.

4.3. Spectral test

The idea is to visually represent the sequence of pseudo-random numbers in 1 dimension (1D), 2D and 3D. For 3D Representation**,** three consecutive values will be the coordinates of a point in space. We look at whether the points are evenly distributed in a cube. By turning the cube as shown in Figure 1, one sees an undesirable effect appear, that is the plans of Marsaglia [20]. It is clear below that the points are located on plans. In fact, all linear congruential generators (LCG) suffer from this effect (this is due to the fact that we do not generate all reals, but only fractions). The smaller the inter-planar distance, the better the generator.

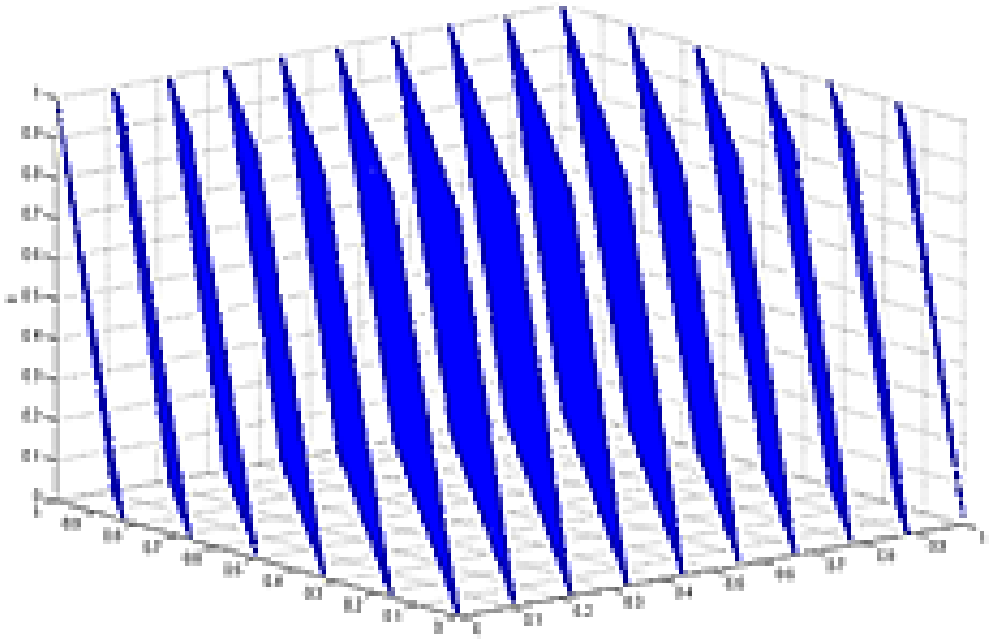

Figure 1. Marsaglia shots

4.4. Poker test

The idea of this test is to compare the theoretical frequencies of hands in poker with the frequencies observed by simulating these hands (a hand is a set of cards). A result can be considered as an ordered list of 4 digits. There are in all 10^4. The theoretical probabilities obtained are as follows:

a) For 4 different digits (eg. 1574), the number of possible cases is 10 * 9 * 8 * 7, 10 for the first number, 9 for the next, and so on the probability is therefore (10 * 9 * 8 * 7) / 10000.
b) For a pair, type ABCC (eg. 4849), we have 10 * 9 * 8 * 1 ways to make it, to multiply by the number of ways to place the pair among the 4 possible places: The probability is: $\binom{4}{2} * \frac{10*9*8}{10^4}$.
c) For a double pair, type AABB (ex: 7337), we have 10 ways to choose the first number, then 9 ways to choose the second one (which must be different), in all 10 * 9 = 90 choices, to multiply by the number of ways to place these numbers among the 4 possible places ($\frac{1}{2} * \binom{4}{2} = 3$), the probability is = $\frac{1}{2} * \binom{4}{2} * \frac{10*9}{10^4}$.
d) For three identical digits, type AAAB (ex: 5515), there are 10 * 9 ways to make it, to multiply by the number of ways to place the three digits among the 4 possible places, the probability is = $\binom{4}{3} * \frac{10*9}{10^4}$.
e) Finally, for four identical digits (ex: 4444), it is worth $\frac{10}{10^4} = 0.001$.

Note that the sum of these five probabilities gives 1.

5. RESULTS AND INTERPRETATION

The results are made under a PC TOSHIBA, on which is installed Windows 7 (32 bits), RAM: 4.00GB, AMD E-450 APU processor with Radeon (TM) HD Graphics 1.65 GHz. The functions developed using MATLAB [23-24] allow us to generate pseudo-random sequences and analyze their performance. Figure 2 illustrates the sequence of pseudo random numbers according to the Von Neumann medial square method, we take as parameter n representing the length of the sequence n = 65536, and the seed x0 = 236589741. After generating the suite, we applied statistical tests to evaluate it.

470554644 , 216729899 , 184912055 , 246808432 , 440210629 , 853978845 , 798677075 , 850701305 , 927103287 , 205047662 , 454369166 ,
513390115 , 694101797 , 773045986 , 964707 , 306595958 , 108146193 , 559906039 , 947725086 , 828386337 , 239233282 , 256321649 ,
78774607 , 543870800 , 954470926 , 147485792 , 205884186 , 829804488 , 754883049 , 484176675 , 270526140 , 439242329 , 338235853 ,
34922546 , 958421912 , 725614017 , 157016668 , 423402982 , 700851664 , 930549315 , 220276469 , 172279510 , 22956584 , 4748949 ,
552516604 , 745976956 , 816188830 , 642062167 , 438262927 , 743931826 , 345617356 , 513567684 , 517660491 , 723839423 , 435102889 ,
145240161 , 470436730 , 107169330 , 526529264 , 330658483 , 350323798 , 267634451 , 819936207 , 953835495 , 21515218 , 904605587 ,
112680316 , 685361385 , 202280491 , 739703920 , 618892633 , 280911816 , 144836841 , 771051085 , 197756796 , 775036418 , 814492262 ,
976448578 , 518254782 , 880190658 , 355944304 , 963475500 , 850391002 , 648562825 , 337379719 , 250747925 , 452189180 , 750545090 ,
179321231 , 610388735 , 744078149 , 522918192 , 434355245 , 644788590 , 523257941 , 988728195 , 834435879 , 832361625 , 258747726 ,
38571017 , 772335241 , 17244905 , 386748459 , 743705388 , 977041402 , 99012221 , 341990735 , 576628258 , 1479241 , 881539360 ,
116432292 , 647862037 , 252189857 , 972397368 , 566412933 , 236106696 , 637189603 , 105901712 , 517260453 , 583762377 , 785128006 ,
259858055 , 620874838 , 855644615 , 277071784 , 877348894 , 410818030 , 714537730 , 641675935 , 480055581 , 533608492 , 380227345 ,
728338857 , 774906160 , 795568059 , 285365010 , 318893230 , 928921398 , 949636622 , 97138435 , 587555424 , 213762718 , 449960674 ,
646081465 , 212594165 , 627899204 , 574103838 , 952168063 , 240201971 , 698687228 , 638425703 , 873782510 , 958747819 , 973804372 ,
949549263 , 438028638 , 690877081 , 111410510 , 230173846 , 999938243 , 764898139 , 691630456 , 526876667 , 990222290 , 401836128 ,
722737660 , 497251822 , 593744823 , 329148393 , 386646144 , 952406700 , 785222048 , 736646653 , 482913761 , 57005631 , 964196570 ,
750255997 , 840610344 , 257504397 , 851447433 , 627311622 , 198710962 , 604641896 , 918223984 , 352847928 , 16602938 , 657550231 ,
723062881 , 199298800 , 1168144 , 645604047 , 45855027 , 268350117 , 178529391 , 274345083 , 522456627 , 609270962 , 111051364 ,
240544626 , 171709747 , 423721480 , 398926133 , 420595903 , 9136203 , 470205257 , 929837104 , 970399751 , 756767408 , 969098110 ,
511468055 , 995712854 , 440876208 , 718307804 , 661012873 , 380182717 , 388983055 , 78170771 , 66943873 , 148213224 , 715976847 ,
228454400 , 141287936 , 228085914 , 318416521 , 890808457 , 397070627 , 650828261 , 774253162 , 679588669 , 407590331 , 298779246 ,
903784032 , 255764981 , 572550593 , 141815446 , 162072417 , 746835222 , 628488197 , 974137683 , 442254406 , 889596264 , 815129227 ,
356567096 , 400939498 , 524810564 , 261280859 , 768727977 , 427026225 , 513968377 , 634925560 , 304667413 , 223254411 , 253203095 ,
180731757 , 396798830 , 493114893 , 622976984 , 3225937 , 406669527 , 801041904 , 681319639 , 964504870 , 696442537 , 322073429 ,
312936678 , 936443767 , 269287531 , 577435207 , 314182831 , 85129517 , 703466465 , 650673795 , 763874997 , 50110417 , 105389191 ,
688157963 , 613820403 , 754871390 , 308154405 , 913732090 , 63322957 , 979688322 , 892082631 , 114205318 , 285465948 , 80746753 ,
3812004 , 531374496 , 588549992 , 910930832 , 949806882 , 331130945 , 477027365 , 551069588 , 776908184 , 863263661 , 241484031 ,
453722800 , 643792398 , 686517225 , 59002217 , 126161091 , 662088231 , 608256287 , 757106750 , 106308955 , 159391319 , 559257255 ,
686772701 , 567428388 , 749755082 , 326829848 , 177495437 , 463015582 , 834291747 , 427191123 , 922555700 , 90196024 , 532274540 ,
161859322 , 844011829 , 559674919 , 360149576 , 77170929 , 535228272 , 693031481 , 926336570 , 994409193 , 496431229 , 439651264 ,

Figure 2. Random data suite

5.1. Test 1: mean, standard deviation and auto-correlation factor

Table 1 illustrates several tests with the same seed and of different value n:

Table 1. Mean, standard deviation and auto-correlation factor

Test	Seed	n	Mean	Auto-Correlation	Variance
1	5746	5000	0.521748	0.272404	0.052689
2		10000	0.521797	0.272363	0.052520
3		20000	0.521821	0.272343	0.052435
4		65336	0.521838	0.272329	0.052376
5	236589741	5000	0.509834	0.260258	0.082758
6		10000	0.507588	0.257629	0.083097
7		20000	0.504481	0.254248	0.083054
8		65336	0.504032	0.254105	0.083523
9		65336	0.494660	0.244056	0.083462
10	236589741	5000	0.509834	0.260258	0.082758
11	236684741	6000	0.501728	0.251207	0.083300
12	289784741	7000	0.496664	0.247977	0.084119
13	236589741	5000	0.509834	0.260258	0.082758
14		6000	0.509984	0.259960	0.083188
15		7000	0.511130	0.260598	0.083012

Note that Test 9 corresponds to the standard sequence of the continuation of Test 8. According to the results obtained and after several tests, it has been found that with n sufficiently large, and with sufficiently large seed (of large digit or > 6) the calculated values gradually tend towards the ideal as shown in Figure 3 (Test 8).

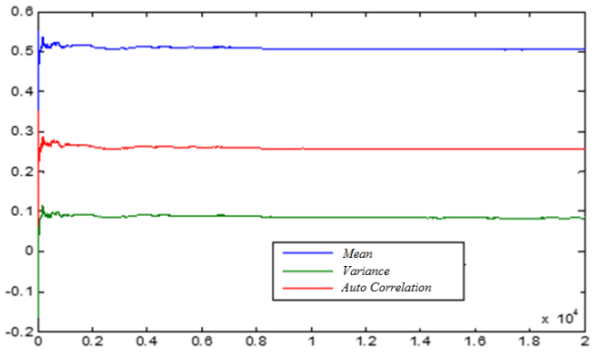

Figure 3. The graphical curve of the three calculated values

5.2. Test 2: the spectral test

This test aims to analyze the distribution of points in 2D and 3D space, Figure 4 illustrates the visual representation of the sequence generated (Test 8). In the 3D representation, by rotating the cube, one visually notices the absence of the Marsaglia planes and the points are uniformly distributed, this implies that the tested suite is random.

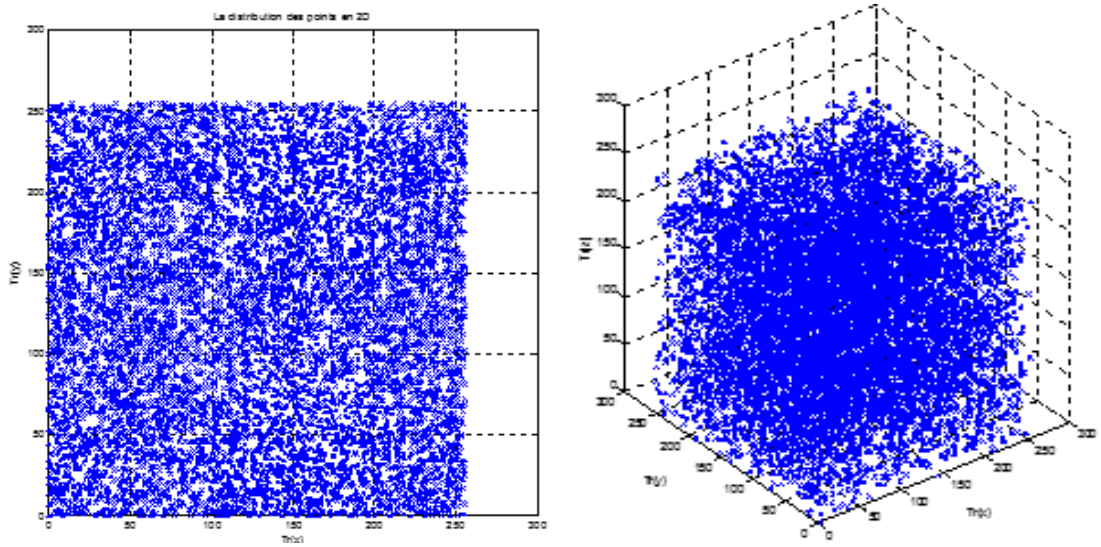

Figure 4. Distribution of values in 2D and 3D

5.3. Test 3: frequency test

Recall that this test evaluates whether the sequence is random by comparing the calculated P-value with the significance threshold α (taking α = 0.01). If P-value > α, then the sequence is random otherwise it is not random. The standardized P-value column corresponds to the normalized sequence (in mod 256). The results are presented in Table 2.

Table 2. Frequency test results

Test	n	Seed	P-value	P-value (normalized)	Interpretation
1	5000	236589741	0.9251	0.7878	+
2	10000		0.9885	0.5637	+
3	20000		1	0.3806	+
4	65536		1.5403	0.0447	+

Note that for all four tests, the P-value is greater than α, so we accept the null hypothesis which states that "the sequence is random".

5.4. Test 4: entropy test

In Table 3 we find the entropy values for the normalized sequences with different n, the higher the entropy, the more random the sequence is. In the ideal case the entropy value is equal to 8. Note that each time the value of n is increased, the value of the entropy also increases.

Table 3. Entropy test results

Test	n	Seed	Entropy (normalized)
1	5000	236589741	7.9628
2	10000		7.9793
3	20000		7.9899
4	65536		7.9934

5.5. Test 5: poker test

The purpose of this test is to compare the theoretical probabilities Pt of poker hands with the probabilities observed Po by simulating these hands (a set of cards). Noting here that the Po are obtained by dividing the observed frequencies of each case on the total number n.

5.5.1. For 4's hand

This time we calculate the theoretical probabilities, with k = 4 shown in Table 4.

Table 4. Theoretical probabilities for hand poker of 4

	4 different cards	1 pair only	2 distincts pairs	3 identical cards	4 identical cards	n
Pt	0.504	0.432	0.027	0.036	0.001	
	0.4600	0.4400	0.0600	0.0300	0.0100	100
	0.0092	0.9888	0.0012	0.0006	0.0002	5000
Po	0.0046	0.9944	0.0006	0.0003	0.0001	10000
	0.00023	0.9972	0.0003	0.0001	0.0001	20000
	$7.019*10^{-4}$	0.9991	$9.155*10^{-5}$	$4.577*10^{-5}$	$1.525*10^{-5}$	65536

5.5.2. For 3 hand

The same principle is used to calculate the theoretical probabilities, but this time with k = 3. In this case the test is applied to the standard sequence or the number of digits reduced to 3 shown in Table 5.

Table 5. Theoretical probabilities for hand poker of 3

	3 different cards	1 pair only	3 identical cards	n
Pt	0.7200	0.2700	0.0100	
	0.6400	0.3600	0.0200	100
	0.6982	0.2898	0.0120	5000
Po	0.7042	0.2846	0.0112	10000
	0.7014	0.2864	0.0122	20000
	0.7032	0.2844	0.0124	65536

5.6. Encrypting images

Using the MATLAB software, we have developed an application to encrypt and decrypt an image, using the values produced by the middle square generator, associated with the XOR symmetric encryption technique. The XOR '1' digit handles the bits, based on the operation or exclusive bitwise (XOR) as shown in Figure 5.

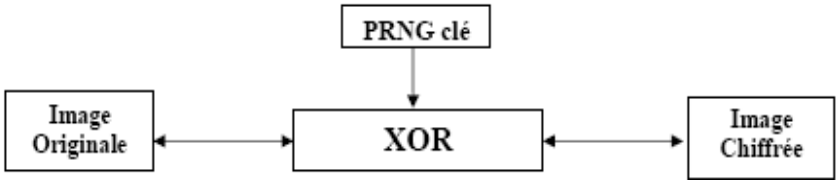

Figure 5. Principle of encryption

We use as a key a bit string K of given length L. It is encrypted by performing the exclusive-or operation bit by bit of the key K with the clear text, divided into blocks M of length L each. Recall that the exclusive-or is associative, commutative and, that it has a neutral element 0, and that any chain K is its own inverse: K⊕K = 0. Thus, we can see that the decryption algorithm is identical to the encryption algorithm, with the same key:

$$M = C \oplus K = (M \oplus K) \oplus K =$$
$$M \oplus (K \oplus K) = M \oplus 0 = M$$

The basic idea of this process is to perform an "exclusive" or "⊕", bit by bit between the key generated by a PRNG, and the image to be encrypted, IO. This algorithm is completely symmetrical, that is the same operation is applied again to the encrypted image, IC to find the original image. The images used in our application are as in Figure 6 and of size 256x256.

We present in the following Figures 7-14 images of the two images "EI: Innocent Children" and "FA: Algerian Woman" according to different seeds to see the influence of the size of the seed in this cipher. Note that the encryption of the 2 images has failed with sequels that have a seed of 2, 3, 4, 5 or 6 digits. From these results we can conclude that to quantify an image it is necessary that the seed is sufficiently wide (large digit is greater than 6). This confirms the condition made by Von Neumann using 10-digit numbers.

Figure 6. IO-EI – IO-FA

Figure 7. IC-EI – IC-FA with seed=26

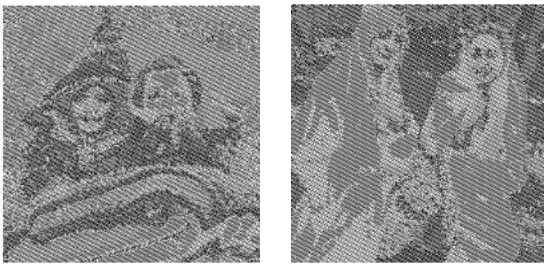

Figure 8. IC-EI – IC-FA with seed=684

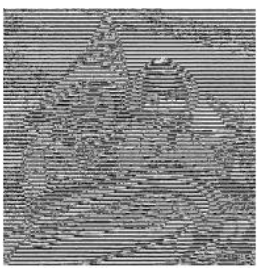

Figure 9. IC-EI – IC-FA with seed=2863

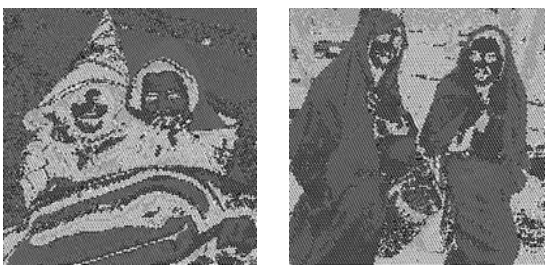

Figure 10. IC-EI – IC-FA with seed=43295

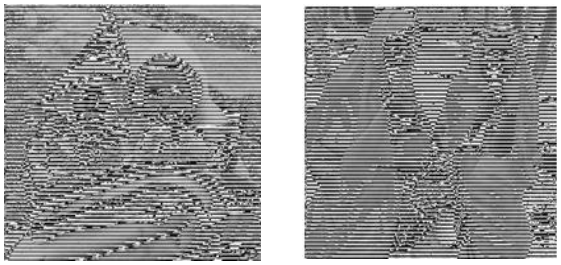

Figure 11. IC-EI – IC-FA with seed=928763

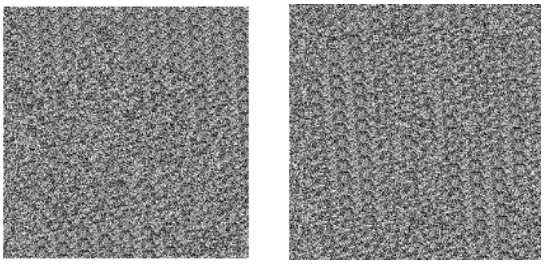

Figure 12. IC-EI – IC-FA with seed=8547267

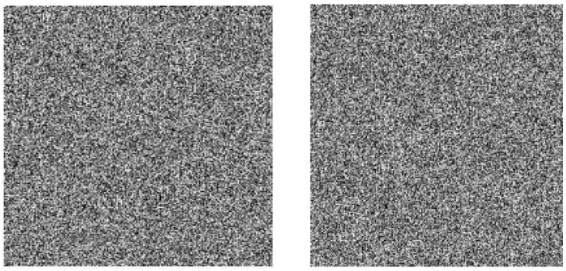

Figure 13. IC-EI – IC-FA with seed=59412678

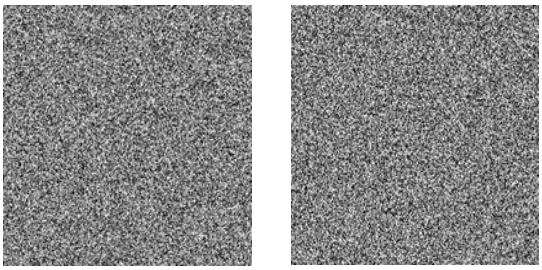

Figure 14. IC-EI – IC-FA with seed=236589741

5.7. Analysis of histograms

A histogram is a statistical curve indicating the distribution of the pixels of an image according to their value. In our work, processed images are grayscale images whose pixel values vary in the range [0, 255]. We have drawn and analyzed the histograms of the encrypted images of "EI- Innocent Children" and "FA-Algerian Women", the illustrated plots the histograms of the encrypted images, HC of the images EI and FA respectively according to the size of the seed. Note that, the historgram for the plaintext is denoted as HP.

It thus emerges from the preceding results that the histograms of Figures 21-23 are uniformly distributed with respect to the histograms of the original images from Figure 15 with respective encrypted images from Figures 16-20 (there is a leakage of information about the pixel distribution of the images). This makes cryptanalysis increasingly difficult because the encrypted images provide no element, based on the exploitation of the histogram, to design a statistical attack on the proposed image encryption technique.

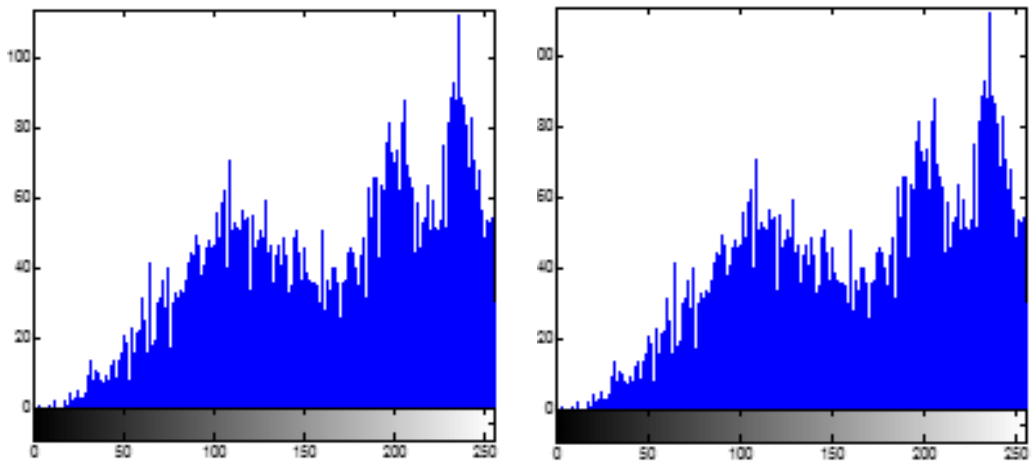

Figure 15. HP-EI – HP –FA

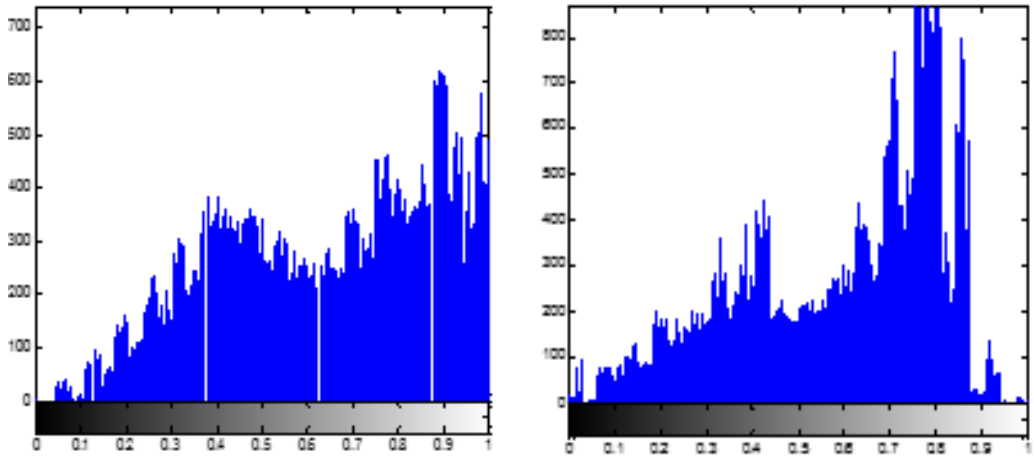

Figure 16. HC-EI – HC-FA with seed=26

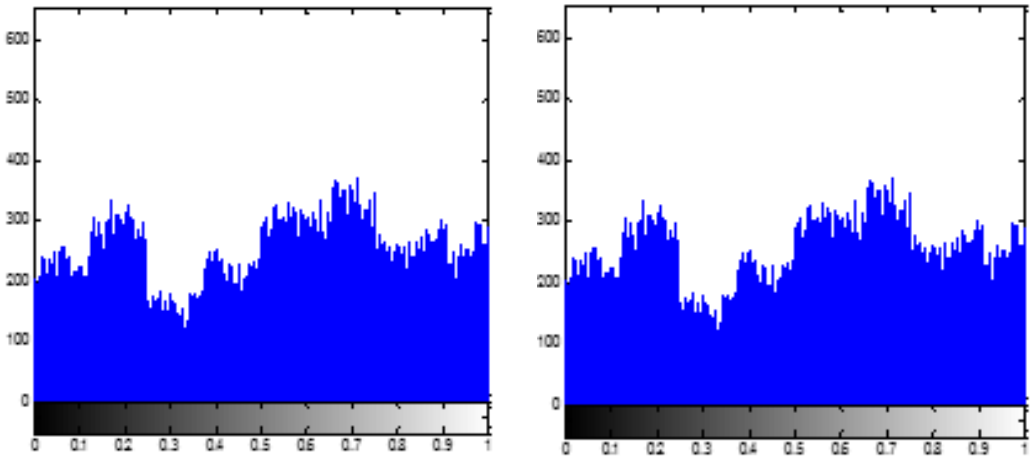

Figure 17. HC-EI – HC-FA avec seed=684

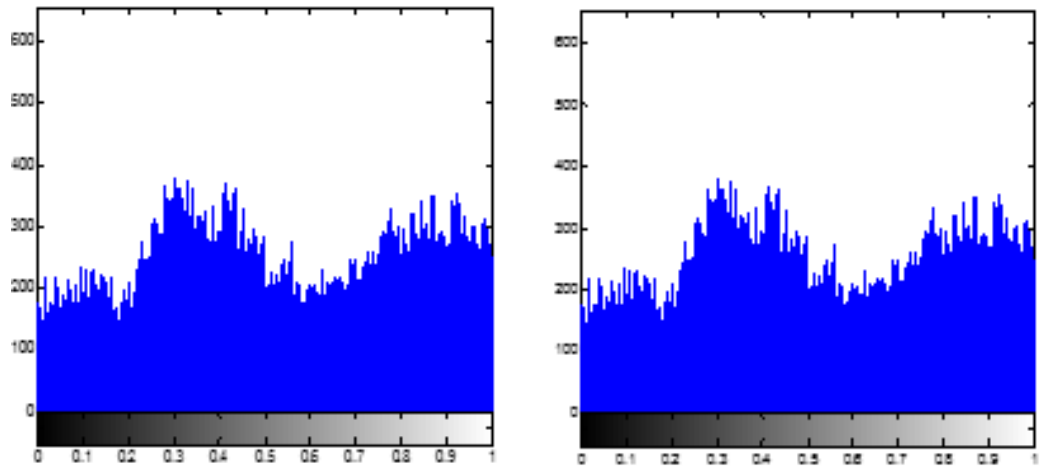

Figure 18. HC-EI – HC-FA with seed=2863

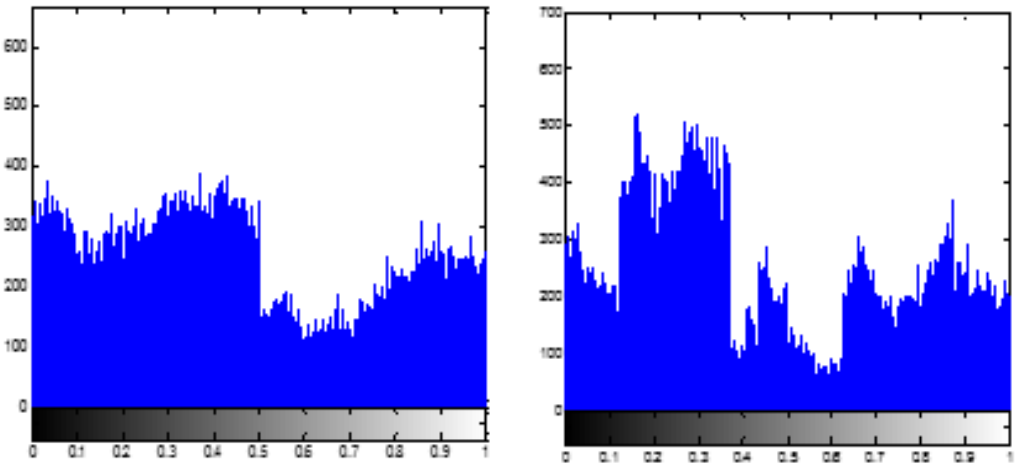

Figure 19. HC-EI – HC-FA with seed=43295

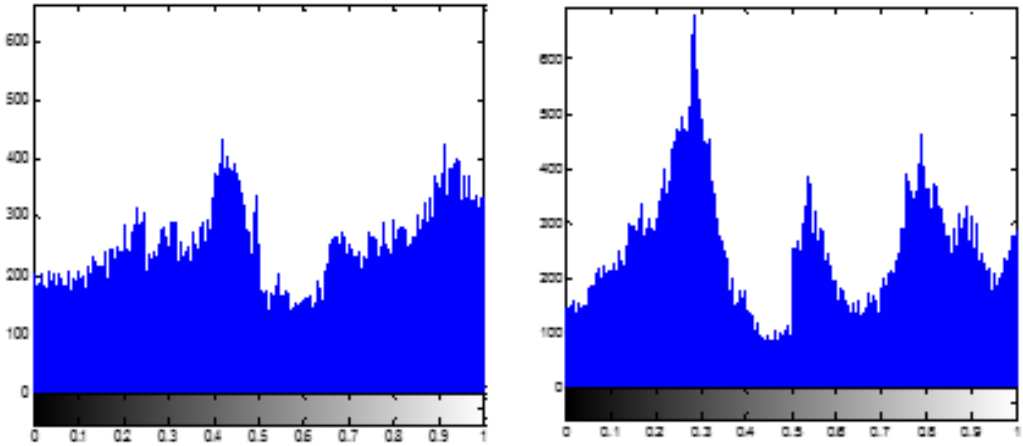

Figure 20. HC-EI – HC-FA with seed= 928763

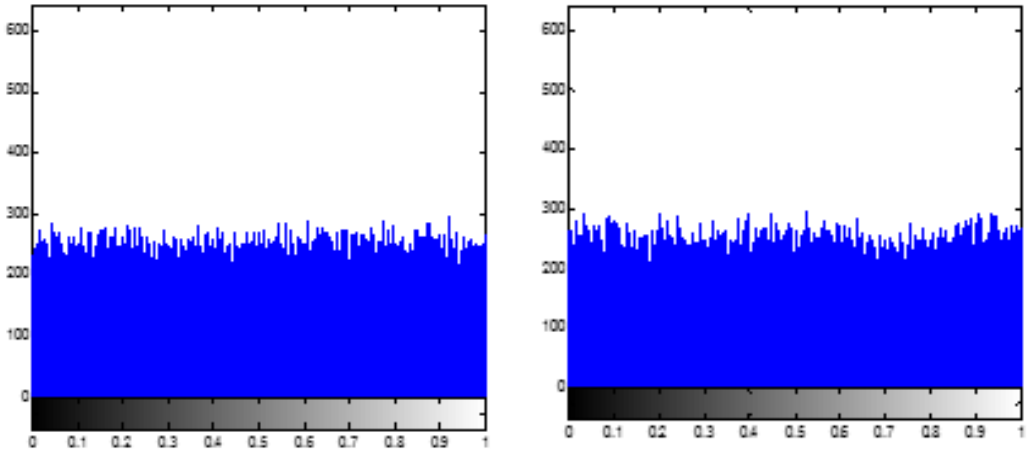

Figure 21. HC-EI – HC-FA with seed= 8547267

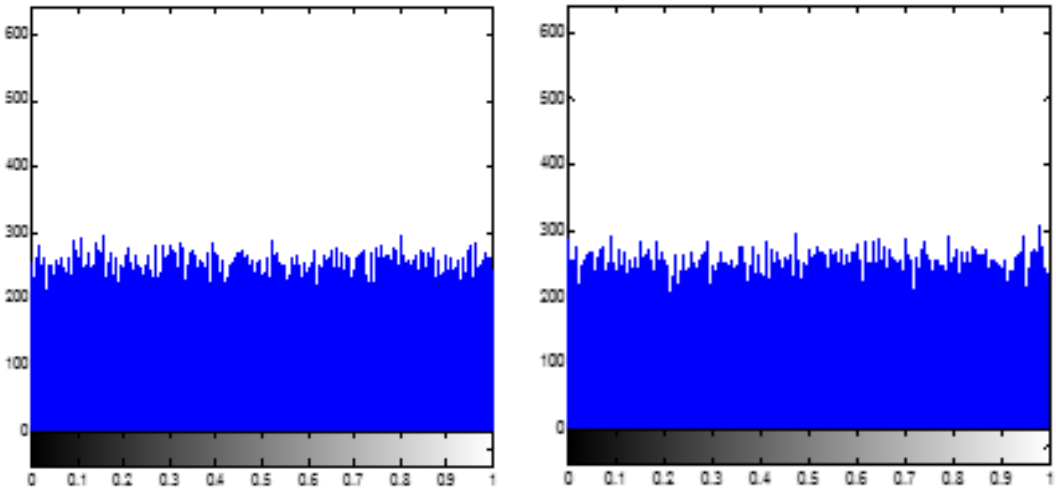

Figure 22. HC-EI – HC-FA with seed= 59412678

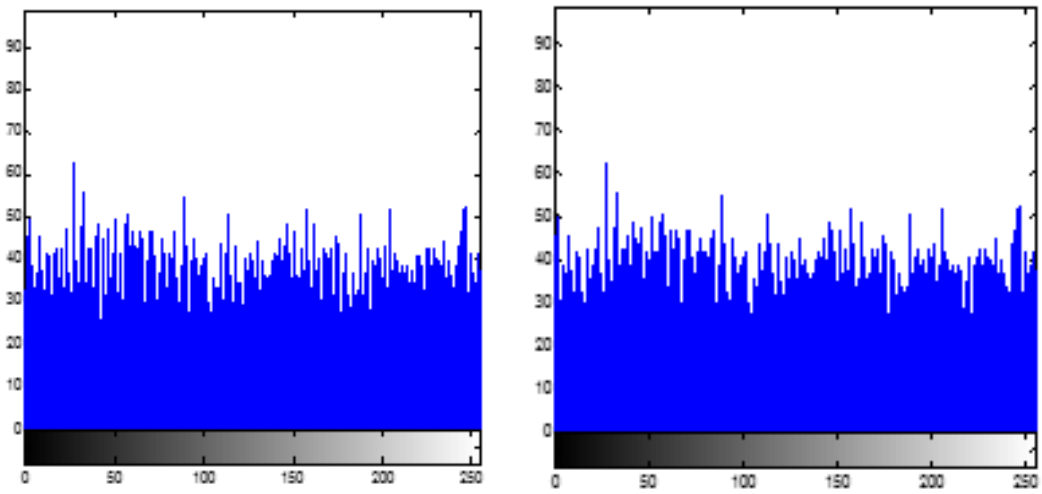

Figure 23. HC-EI – HC-FA with seed= 236589741

This explanation is reinforced by Table 6 associated with the entropies of the different encrypted images of the two images "EI: Innocent Children" and "FA: Algerian Woman". In addition, this table informs us about the quality of the images for Figures 8-14.

Table 6. Entropy of encrypted images

Test	Seed	EI	FA
1	0	7.6916	7.4616
2	26	7.6916	7.4616
3	684	7.9676	7.9221
4	2863	7.9646	7.8748
5	43295	7.9348	7.8525
6	928763	7.8716	7.8716
7	8547267	7.9970	7.9964
8	59412678	7.9975	7.9968
9	236589741	7.9972	7.9975

6. CONCLUSION

In this work we have adapted Von Neumann's middle square generator to digital image encryption and to real-time application. We first started by analyzing the performance of the number sequences produced by the middle square generator and explain how to interpret the results, then we discussed the principle of encrypting the images with the keys generated using the encryption system continuously with the XOR operation, to finally analyze the encrypted images and validate the algorithm. The size of the key used, the power of the algorithm and the ability to keep keys secretly secure, determine the robustness of an encryption system. One of the main characteristics of the encryption algorithms studied is that they make it possible to achieve a very high level of performance. These performances are expressed either in terms of speed of encryption or in terms of material efficiencies. Our middle square generator with its simplicity could verify some quality criteria, since it satisfied some tests (in the majority of the cases) with success and it managed the encryption of the images.

ACKNOWLEDGEMENTS

The authors thank the Government of Malaysia for funding this research under the Fundamental Research Grant Scheme (FRGS/1/2018/ICT03/UNISZA/02/2) and also Universiti Sultan Zainal Abidin, Terengganu, Malaysia.

REFERENCES

[1] B. Schneier, *Applied cryptography*, John Wiley & Sons, Inc., 1994.
[2] R. L. Rivest, *et al.*, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, vol. 21, pp. 120-126, 1978.
[3] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computations*, vol. 48, pp. 203-209, 1987.
[4] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Proc. CRYPTO 1985, LNCS 218*, pp. 417-426, 1985.
[5] M. A. Mohamed, "A survey on elliptic curve cryptography," *Applied Mathematical Sciences*, vol. 8, pp. 7665-7691, 2014.
[6] A. Sambas, *et al.*, "A novel chaotic hidden attractor, its synchronization and circuit implementation," *WSEAS Transactions on Systems and Control*, vol. 13, pp. 345-352, 2018.
[7] H. Ali-Pacha, *et al.*, "An efficient schema of a special permutation inside of each pixel of an image for its encryption," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 11(2), pp. 496-503, 2018.
[8] J. I. Ahmad, "Analysis Review on Public Key Cryptography Algorithms," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 12(2), pp. 447-454, 2018.
[9] N. Rajitha and R Sridevi, "Cryptographic Key Protection in a Cryptoprocessor," *International Conference on Information Security & Privacy*, Nagpur India, pp. 271-275, 2015.
[10] B. A. Muzakkari, "Recent advances in energy efficient-QoS aware MAC protocols for wireless sensor networks," *International Journal of Advanced Computer Research*, vol. 8, pp. 212-228, 2018.
[11] M. Soucarros, "Analysis of random number generators under abnormal conditions of use," PhD Thesis, University of Grenoble, discipline: Mathematics, 2012.
[12] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, pp. 337-343.
[13] M. A. Mohamed, *et al.*, "An empirical analysis of six pseudo-random number generators," *Far East Journal of Electronics and Communications*, vol. 17, pp. 1373-1388, 2017.
[14] S. K. Park and K. W. Miller, "Random Number Generators: Good Ones Are Hard to Find," *Communications of the ACM*, vol 31, pp. 1192-1201, 1998.
[15] P. Gayathri, *et al.*, "Hybrid Cryptography for Random-key Generation based on ECC Algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7(3), pp. 1293-1298, 2017.
[16] M. L. Pseudo, *Randomness and Cryptographic Applications*, Princeton University Press, 1996.
[17] P. Dąbal, "Pipelined pseudo-random number generator with the efficient post-processing method," *International Journal of Microelectronics and Computer Science*, vol. 6, pp. 43-48, 2015.
[18] M. Bakiri, *et al.*, "Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses," *Computer Science Review*, vol. 27, pp. 135-153, 2018.
[19] Y. Dodge, "A Natural Random Number Generator." *International Statistical Review/Revue Internationale De Statistique*, vol. 64, pp. 329-344, 1996.
[20] B. F. Vajargah and R. Asghari, "A Novel Pseudo-Random Number Generator for Cryptographic Applications," *Indian Journal of Science and Technology*, vol. 9, pp. 1-5, 2016.
[21] L. E. Bassham, *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," *NIST Special Publication* (800-22 Rev 1a), 2010.
[22] U. M. Maurer, "A Universal Statistical Test for Random Bit Generators," *Journal of* Cryptology, vol. 5, pp. 89-105, 1992.
[23] S. J. Chapman, *Essentials of MATLAB Programming*, Second Edition, Cengage Learning, 2008.
[24] D. E. Knuth, "The Art of Computer Programming," Seminumerical Algorithms, Addison-Wesley, vol. 2, 1981.