Journal of Theoretical and Applied Information Technology
15th April 2018. Vol.96. No 7
© 2005 – ongoing JATIT & LLS

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

SURVEY PAPER: PSEUDO RANDOM NUMBER GENERATORS AND SECURITY TESTS

[1]OMAR SALHAB, [2]NOUR JWEIHAN, [3]MOHAMMED ABU JODEH, [4]MOHAMMED ABU TAHA, [5]MOUSA FARAJALLAH

[1,2,3,4,5]College of Information Technology and Computer Engineering (CITCE)
Palestine Polytechnic University, Palestine

E-mail: [1]131082@ppu.edu.ps, [2]131035@ppu.edu.ps, [3]131089@ppu.edu.ps, [4]m_abutaha@ppu.edu,
[5]mousa_math@ppu.edu

ABSTRACT

Many security applications are based on Pseudo Random Number Generators (PRNGs). Random binary numbers constitute a major reliance in many network security algorithms. For example, common cryptosystems require a long dynamic key that should be generated from a short secret key and has the random behavior. Random or pseudorandom inputs are critical requirements in many protocols that need this issue at certain points. A PRNG is a deterministic algorithm generates a sequence of bits simulates the truly random numbers sequence behavior. Each generated number should be independent of the previous or the future numbers, as a result the PRNG become unpredictable. However there are many security and performance tests can be applied on the PRNG sequence to evaluate it. And then measure the power of the PRNG. Therefore not all PRNG are good enough to be used in cryptographic applications. This depends on the kind of application and its data sensitivity. A PRNG that passes these tests can be considered as a secure PRNG. Furthermore, it can be used in many cryptographic applications. In this survey, some missing results are reproduced by our team in order to have the same level of assessment for presented algorithms under the test. Moreover, new test tools are used to evaluate the behavior of the generators and assess the randomness of them. Finally, details discussions of the new tools are considered in order to validate the security level of the proposed generators.

Keywords: *PRNG, Stream Cipher, RC4, Salsa20, NIST Tests*

1- INTRODUCTION

Because randomness considered as the Foundation stone to the cryptographic systems, any adversary can see the final result of system as a sensitive data included in a random number sequence without any indication about the real information [1].

There are many mechanisms in cryptography science to provide solutions for various security cases; most of these techniques need randomness for many different reasons.
Therefore, there are many conditions should be taken into consideration during study and analyze of randomness, and that is what we have tried to do in our survey

First of all, a set of basic concepts that will help to provide a good description of PRNG is reviewed. In cryptography science, Encryption is the process of transform original message (Plain Text) to non-readable data (Cipher Text) using an encryption algorithm. This Cipher Text can't give anyone any information about the Plain Text except those who have the encryption key.

A simple encryption example can be performed by replacing every character in the data with its next character, so the word “survey” will be encrypted to “tvswfz”. There are two main types of encryption: Asymmetric cipher, and Symmetric cipher [2-4], as shown in Figure 1.

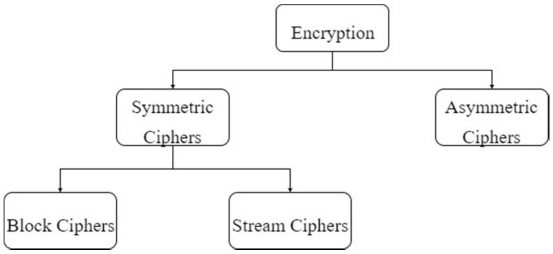

Figure 1. Encryption Models

Asymmetric cipher, also called a public key encryption, an encryption technique which uses a pair of public key and secret key. The sender has the public key of the receiver while the secret key is not known. The receiver's should create his pair of the public and secret key, publish his public key without considering its security. The secret key should be computational impossible to find through the public key. Asymmetric encryption is used in authentication and digital signatures. A signed message with the sender's secret key proof the identity of the sender and it can be read by anyone who the sender's public key. Thus, the receiver can ensure that the message has not been modified or replaced by any other source which is confirms the sender identity [5-6].

The second type of encryption is called Symmetric cipher [6-8]. In this type, both sender and receiver shared the same secret key. It uses in the encryption and the decryption process. In symmetric ciphering, the receiver and sender share the same secret key. Symmetric is faster than a symmetric one but it has a lower security level.

Figure 2 shows the general structure of this encryption model.

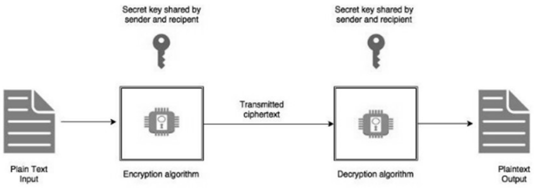

Figure 2. Simple Symmetric model

As it is shown if Figure 2, the Symmetric ciphers can be used as block cipher or stream cipher.

1) **Block cipher**: the plaintext is divided into number of blocks, the block size depends on the specification of the used encryption algorithm [9]. The divided blocks are encrypted as one unit and there is a relationship and dependency between encrypted blocks based on the encryption mode. This type of encryption has better security than the stream cipher against number of well-known attacks, moreover the most important properties of the secure cipher text which are the confusion and the diffusion properties are included inside block ciphering algorithms. . Where in terms of execution time it is slower than the stream cipher and so the encryption throughput of stream cipher is much higher than the block cipher.

Stream cipher: in this type, the encryption is performed bit by bit, and the most important the encryption of each bit is independent of other bits and so the diffusion and confusion properties are not achieved in this type. The encryption operator is as simple as possible, and in most case is the XOR operation between the plaintext bits and the corresponding key bits. In terms of encryption throughput (speed of encryption) the stream cipher is much more than the block cipher [10].

However, the key should be randomly generated, in addition to it should be at least equal to plain text in length. Furthermore, it cannot be used more than once to prevent the two time pad attack. This technique is called the one-time pad.

Random Number Generator (RNG) is a functional technique to generate the key stream from an initial seed. The RNG can be defined as a function that can give a random numbers sequence. There are two types of RNGs; the first is a **True Random Number Generator (TRNG)**. This kind of generator relies on a physical system. It generates a sequence of truly random numbers with no pattern, unpredictable, and with no dependency. Both noise and electric current are familiar sources of TRNG. However, TRNG can not be used in real live application, since the generated numbers at the sender part will be differed from that at the recipient part. The solution for this problem is achieved using deterministic function where pseudorandom, number generators is required [11-12].

Pseudo Random Number Generator (PRNG)

A PRNG is a generator that simulate the random behavior, this generator as any function has number of inputs and produces a pseudorandom sequence of numbers. The inputs of the PRNG are called seeds. This seed should be random and unpredictable. PRNGs are generally used in cryptographic applications. For example, generators can be used in session key generation, to be used during the stream cipher encryption as a generated dynamic keys [13]. PRNG generated bits should have the randomness and unpredictability [14-15].

1. **Randomness:** Each individual random bit has a probability of exactly 0.5 of having a value of a “0” or a “1”. Assess and evaluate the sequence regarding the random based on: independence and uniform distribution of generated bits inside the sequence. The generated bits are independent of each other, bit doesn’t depend on the previous or the future ones. The bit distribution inside the generated sequence should be followed the uniform distribution. There exist well-defined statistical tests to assess if the generated sequence follow the uniform distribution or not. However, there are no statistical tests to “prove” the independency [16].
2. **Unpredictability:** Random and pseudorandom generated bits should be unpredictable in order to be used in cryptographic applications. The non-authenticate user should not be able to predict or calculate the next or the previous bit from current one with non-negligible guess [15-16].

Some PRNG algorithms are more simple and faster than others as they are based on simple equation. However, other algorithms may contain more complicated equations, so they cost more space and time. There are also some weaknesses in PRNGs, such as getting in a loop, having a small period and high complexity. In this survey we will review some stream cipher algorithms, mainly RC4 and Salsa20 which are wildly used in stream cipher. In addition, the National Institute of Standards and Technology (NIST) tests will be applied to the random binary sequences in order to verify the security of the proposed generators.

PRNG Challenges

Some aspects should be overpassed for the PRNG to be suitable for cryptographic applications. These aspects considered as challenges, hereafter we list some of them [17].

1) **Correctness**: if the output is random and is not predictable.
2) **Reliable**: does not fall into loops often or fail.
3) **Performant**: produces random behavior numbers with a fast bitrate.
4) **Efficient:** the usage of power is considered.
5) **Affordable:** does not cost much.

In this survey paper, the most important tests and tools used to evaluate the behavior of random number generators are highlighted, as well as using a new methodology for evaluation and assessment. The intent of this paper is to introduce the reader to the testing of random generators. In doing so, we have discussed some definitions of randomness, different types of random number generators, and applications for these numbers. In the following sections we will introduce the reader to the various statistical tests and provide him with the required knowledge to be able to apply these tests himself. It is worth mentioning that in previous researches, such as L'Ecuyer’s paper [18], similar work has been conducted. However, it wasn’t in the same level of assessment as ours. Our paper comprises the most recent and important set of statistical tests and tools. One might ask: why are there so many tests? Well, a bad generator can pass some tests, so trying more than one test is essential to make sure that a bad generator is detected as such. For example, the non-random sequence: 1010101010101010101010101010 passes the frequency test with a perfect score (i.e. number of zeros and ones are exactly the same), but fails miserably in the serial test. This is why it’s important to apply various statistical tests to the sequence. Furthermore, we presented the results obtained from applying the statistical tests over sets of data obtained from various ciphers, such as Salsa20 and RC4. However, our paper will not include the math models for the various generators presented. Known attacks that are not based on statistical attacks will not be included too as the focus of our paper will be on the statistical analysis of generators.

The remainder of this document is organized as follows. Section 2 presents few previous researches that are done in this field, more specifically, some generators, RC4, Salsa20, HC-128 and HC-256, and SOSEMANUK models. The NIST test suite, visualization, Histogram, Chi-square, Mapping, and Correlation tests are presented and summarized in Section 3. We did some statistical tests for some ciphers and the results will be reviewed in Section 4. Finally, a comparison between some stream ciphers in terms of speed and performance is carried out as well in Section 4.

2. RELATED WORK

This section reviews some of the states of the art researches that are proposed in the field of PRNG. Section 2.1 reviews some proposed PRNGs, while section 2.2 reviews the standard ciphers along with the eSTREAM ciphers.

2.1 PRNGs

In this section, we reviewed four state of the art PRNGs.

2.1.1 Cryptosystem based on an efficient chaotic generator

Abutaha et al [19] proposed a stream cipher cryptosystem based on an efficient chaotic generator of finite computing precision, where N=32. The proposed is composed of an "IV-Setup, a Key-Setup, a non-volatile memory, an output and an internal state function" [19]. It uses the internal feedback mode, where the generated keystream is used in stream cipher. The internal state function contains two recursive filters. The first recursive filter based on the discrete skew tent map, while the second recursive filter based on the discrete piecewise linear chaotic map. Each of the two recursive filters using the perturbation technique based on the well-known linear feedback shift register (LFSR). The stream cipher has two implemented versions: sequential and parallel, and they are implemented using Pthread library. In sequential implementation, the generator produces 32-bits of keystream, which will be converted to 4 bytes and stored in a buffer. The 4 generated bytes will then be XOR-ed with 4 bytes of plaintext to produce 4 bytes of ciphertext, and so on. In parallel implementation, the generated produces four 32-bit samples, which are converted to 16 bytes. The generated 16 bytes will be XOR-ed with 16 bytes of plaintext to produce 16 bytes of ciphertext. The parallel implementation of the proposed faster than the eSTREAMS. Various tests such as the NIST tests, Histogram test, Chi-square test and correlation test were applied to the proposed cipher. Furthermore, the security of the stream cipher was investigated by applying software security tools. The results obtained from the statistical tests and cryptographic analysis indicate the robustness of the implemented stream cipher.

2.1.2 Francois PRNG based on two chaotic maps

Francois et al. [20] in their paper proposed "a new pseudo-random number generator based on two chaotic maps" [20]. An input initial vector is responsible for producing the chaotic maps that will be mixed. In their paper, they develop an algorithm that uses chaotic function which is responsible for generating "multiple pseudo-random sequences. The proposed algorithm uses permutations whose positions are evaluated using a chaotic function that is based on linear congruences. These permutations are stored on the initial vector to produce two chaotic maps" [20]. The chaotic map are XORed to generate one sequence of this generator. The generated sequences ready to serve cryptographic applications. Author of this proposal assume the adaptive size of the key space, simplicity of the implementation, and the security against various attacks are main contribution. However, if an opponent knows the parameters used in the linear congruences, prediction of the subsequent numbers is easy when the adversary knows one of the previous generated numbers [21].

2.1.3 MIXMAX generator

Savvidy, K. G. et al. "proposed a new pseudo-random number generator" [22], named MIXMAX random number generator. It is a matrix-recursive PRNG. The period of the generator is 10^{4682} for matrix size N=256. That means the generator will start repeating itself after 10^{4682} iterations. There is an enhanced version using C code implementation of the generator that was developed by Konstantin Savvidy. The generator works under UNIX, Linux and MacOS devices, it was also tested on ARM architectures. The most usage for this algorithm for Monte Carlo simulations which needs a PRNG for physical complicated simulations. However, since the MIXMAX generator uses matrix multiplication, then we believe that it’s relatively slow compared to other PRNGs.

2.1.4 Novel pseudo-random number generator based on quantum random walks

Yu-Guang and Zhao in their paper [23] investigated the idea of applying quantum computation for constructing PRNGs, as well as "constructing a novel PRNG based on quantum random walks (QRWs). A QRW is a famous quantum computation model. The proposed PRNG is based on the equations used in the QRW. As a result, the PRNG algorithm is relatively simple, and the computational speed is fast" [23]. Furthermore, they applied statistical tests, such as NIST test suite to the proposed PRNG and it successfully passed all tests. The proposed QRWs-based PRNG better than PRNGs based on quantum chaotic maps (QCM) [24]. The advantages include better statistical complexity and recurrence. They attempted to compare QRWs-based PRNGs with QCM-based PRNGs by "numerical simulations and performance in terms of quantifiers based on information theory, recurrence plots, and other randomness tests" [23]. It was concluded in their paper "that the new QRWs-based PRNG can generate a high percentage of good pseudo-random numbers, and these numbers can be used in various applications and it also extends the application scope of quantum computation" [23] [25].

However, since QRNGs are typically based on specialized physical hardware, such as Raman scattering or single-photon sources, we think that the cost and power requirements can be important limitation.

2.2 Stream Ciphers

In this section we will discuss stream cipher algorithms including RC4, Salsa20, HC-128, HC-256, and SOSEMANUK:

2.2.1 RC4

RC4 is a stream cipher algorithm was designed by Ron Rivest in 1987. In the laterite review, RC4 is considered as one of the well-known stream cipher algorithms. RC4 is used in WEP, WPA, and SSL. The RC4 stream cipher has two main components: the Pseudo-Random Generator and the key scheduling algorithm. RC4 includes a state vector S. this vector has 256 bytes: S_0, S_1,…, S_{255} and S_i is initialized to i.

Another vector T is created, and it is also of size 256. A key K is used to shuffle the permutation found in vector S. K is copied to T vector, and if the T vector is not filled yet, K will be copied to the remaining T cells till T is filled. The steps that are previously mentioned are for initialization summarized as follows:

```
$for\ i = 0\ to\ 255$ do
    $S[i] = i$
    $T[i] = K[i\ mod\ keylen]$
end                                    (3)
```

Where (keylen) donates the length of the key. The next step is to scramble S and produce an initial permutation. We will first start with S[0] and through to S[255]. For each byte in S, we will swap its value with another byte also in S using the T vector. The scrambling process requires two indices, i and j which are initialized to zero. The scrambling process is summarized as follows:

```
$j = 0$
$for\ i = 0\ to\ 255$
do
    $j = (j + S[i] + T[i])\ mod\ 256$
    $Swap\ (S[i].S[j])$
end                                    (4)
```

This process will just result in scrambling. The vector S will still contain numbers from 0 to 255, but the numbers will be shuffled. That concludes the KSA.

The next step is called the PRGA or Stream Generation. This process takes as input the key-dependent scrambled permutation vector S that was output from the previous step, and it produces a pseudo-random keystream of bytes. i and j are initialized to 0.

RC4 steps are summarized in [8]

For encryption, the n-bit keystream k is XOR-ed with n-bits of the plaintext to produce n-bits of ciphertext. For decryption, the n-bits of k will again be XOR-ed with the ciphertext to recover the original plaintext [26]. Many studies on the cryptanalysis of RC4 are carried out. Moreover, partial information of the used secret key is gained [25]. More researches are proposed on the weaknesses of the RC4 stream cipher [27-38]. "However, all of these exploit the initial keystream bytes only. If some amount of initial keystream bytes is discarded, then RC4 is considered safe to use" [28] [39]. We can see the statistical NIST Tests results over the set of data produced by RC4 in Table 3.

2.2.2 Salsa20

It is a stream cipher proposed by Bernstein [40]. It is based on simple operations like addition, XOR, basic rotation. The multiplication operation and other time consuming operations are minimized in order to produce a fast stream cipher algorithm. Moreover, this stream cipher is resistant to timing attacks. The main component of Salsa20 is 256-bit hash function. Salsa20 divides the plaintext into 64-bits blocks, it XOR-ing the plaintext bits with the output of the hash where the input are the block number and the key. Salsa20 is presented in details by [42].

Salsa20 has three versions proposed by [43-44]:

1) Salsa20/20, number of encryption rounds are 20.
2) Salsa20/12, where number of rounds are 12 instead of 20.
3) Salsa20/8 where number of rounds are 12 instead of 8.

To make a cipher picture, the Salsa20 generates a 64-byte. The 64 byte of the ciphered data is produced by XOR-ing the 64-byte from the plain block with the 64 byte produced by Salsa20 generator.

It generates a uniform random key and this key never reused for different plaintext. We can notice the statistical NIST Tests results over the set of data produced by Salsa20 in Table 4. Finally, Salsa20 and other versions have researches on possible attacks and weaknesses [45-50]

2.2.3 HC-128 and HC-256

"HC-128 is one of the eSTREAM stream cipher, which consists of two secret tables, each one with 512 32-bit elements. At each iteration they update one element from one of the tables using a non-linear feedback function. Every 1024 steps, all elements in the two tables are updated. At each step, a sample of 32-bit output is generated from the non-linear output function. HC-256 is a new variation that differs from HC-128 in the size of secret tables which is 1024 32-bit elements. All of the elements of the two tables are updated every 2048 steps. At each step, HC-256 produces one 32-bit output [53]. However, in 2010, (Kircanski and Youssef) provide in a differential fault analysis attack on HC-128 in their paper. The attack is based on the fact that, some of the inner state words of HC-128 may be exploited several times without being updated. Consequently, the complete internal state is recovered using about 7968 faults" [50-54] [19]. Many researches are presented regarding security weaknesses and randomness of HCI-128 and HCI-256 [55-60]. The statistical NIST Tests results over the set of data produced by HC-128 are provided in Table 5.

2.2.4 SOSEMANUK

SOSEMANUK is a stream cipher where the size of the key is ranging [128➔256] bits [61]. IV size is 128 bits. SOSEMANUK consists of: a Finite State Machine (FSM) and a Linear Feedback Shift Register (LFSR). The FSM has two registers, each one 32 bit: LFSR send the output to the FSM then the FSM update the memory to output four word at each encryption round. In 2011 Salehani et al made a differential attack on SOSEMANUK [62]. This attack is based on the faults and it requires 6144 wrong output in order to recover the partial states of the used secret key. It has a lot of attacks and evaluation analysis weaknesses regarding security level [63-70]. We can notice the statistical NIST Tests results over the set of data produced by SOSEMANUK in Table 6.

3. STATISTICAL AND VISUALIZATION TESTS

In this section, some aspects of testing PRNGs will be discussed. These generators should satisfy harder requirements than generators used in other applications. Therefore, some statistical and visual security tests are used in order to assess and evaluate the randomness of the generated bits.

3.1 NIST Test Suite

To evaluate the statistical performance of PRNGs, various statistical tests should be applied on the binary sequence. The main target of these tests to measure the relation between the PRNG and the TRNG. The adversary should not have the ability to distinguish between TRNG and PRNG outputs. The National institution of Standards and Technology (NIST) released a suite for testing PRNGs that contains 188 tests including 15 main tests. NIST tests try to find the non-random behavior in the generated bits from the proposed PRNG [71].

3.2 Information Entropy

In addition to the NIST tests, we use the entropy test as an extra test to evaluate the proposed generators under the test. Entropy is the average (expected) amount of information produced by the source. This concept was introduced by Claude Shannon in 1948 [72-73]. Given a robust and unpredictable PRNG, the probability of existence any value (more than one bit) should be exactly equal to the probability of the existence of other values. This test can be applied using the following equation:

$$H(S) = \sum_{i=0}^{Q-1} Pro(s_i) \times \log_2 \frac{1}{Pro(s_i)}$$

Where $H(S)$ is the entropy value for the sequence, and $Pro(s_i)$ is the probability of each value to occur.

3.3 Hamming Distance (HD)

More random behavior test are used, which is a measurement used to measure the differences between two generated sequences. The optimal result of HD is achieved when a small change on the secret key produces a 50% differences between the two generated sequences. This test describes the resistance of any cryptosystem to plaintext and/or the secret key sensitivity attacks. The HD is given by:

$$HD(S_1.S_2) = \frac{1}{|Ib|} \sum_{K=1}^{|Ib|} (S_1(K) \oplus S_2(K))$$

Where $|Ib|$ is the size of the generated bits [74].

3.4 Histogram and Chi-square Test

The generated bits of the any proposed PRNG should have a uniform distribution and it is measured using the well-known histogram. However, histogram is a visual test. A numerical test which is called chi-square test is used to confirm the uniform distribution of the histogram.

$$\chi^2 = \sum_{i=1}^{Nv} \frac{(O_i - E_i)^2}{E_i}$$

Where Nv is the number of bits/bytes under the test, O_i is the frequency of the bit/byte at the position i, E_i is the expected frequency [74-75]. in our test each 8 bit is considered as one level, to calculate the expected frequency which is total numbers of 8 bits in the generated samples divided by the total number of levels of the 8-bits which is 256 level.

3.5 Correlation Test

Some applications like that including image have high correlated data. A robust PRNG should completely remove this correlation. The mathematical models and description of all parameters and scenario are described in details inside the following papers [76-78].

3.6 Mapping Test

Mapping test assess the unknown prediction or calculation of the generated PRNG bits. "One of the characteristics of any generated sequence is the phase space trajectory. It reflects the dynamic behavior of the system" [19]. We plot x(n) and x(n+1) sequences on an xy plane. The system is considered secure, if the signature of the generated sequences is unknown.

4. RESULTS AND ANALYSIS

In this section we will provide some results of different statistical test like (NIST, Histogram, Mapping). First, we applied these tests on some of stream ciphers generators. To evaluate ccomputing performance of the some stream cipher models, we performed some experiments using a two 32-bit multi-core Intel Core (TM) i5 processors running at 2.60 GHz with 16G of memory. This hardware platform was used on top of an Ubuntu 14.04 Trusty Linux distribution.

4.1 NIST Test Suite Results

For each cipher, we produced 100 bitstream, each consist of 1,000,000 bits. The *p-value* is used to test the robustness of each PRNG in terms of statistical attacks. The minimum *p-value* is 0.01. If the result of any test is greater than *p-value* the generated sequence has passed that test. Otherwise, it fails to pass that test. However, running the NIST tests on one sequence is not enough, as we might obtain different results when running the tests on two sequences produced by the same generator. To obtain the most accurate results, we should generate 100 sequences using 100 different keys and run the NIST tests on all generated sequences.

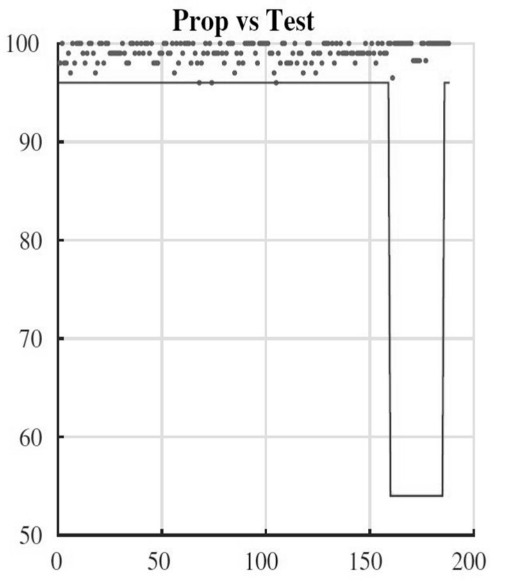

Figure 3. NIST test key stream results For Chaos-based stream cipher

Figure 3 presents the NIST result obtained from the chaos-based stream cipher that was proposed by Abutaha et al [19]. The figure is the result of running the NIST tests on 100 sequences produced by the chaos-based stream cipher. The x-axis represents the subtest number, and the y-axis represents the number of sequences that passed each subtest. The line shown in the figure represents the minimum pass point of each statistical test. It is clear that the minimum pass rate of all statistical tests excluding one test is 96 [79]. The minimum pass rate for the random excursion (the excluding test) test is 59. That is, the number of sequences that need to pass each test should be bigger than or equal to 96, given that the total number of generated sequences is 100. Since most values lie above the minimum pass rate line, then we conclude that the chaos-based stream cipher has high randomness.

Tables 1, 2, 3, 4, 5 and 6 present the results of applying the NIST test suite on the generated bits of the mentioned PRNGs and eSTREAM stream cipher generators. All implemented ciphers have passed the NIST tests with good *p-value* results. The eSTREAM ciphers (Salsa20, SOSEMANUK and HC-128) have good random results in terms of NIST test suite. The NIST results obtained from Salsa20 were better than those produced from RC4, as depicted in Figure 4 and Figure 5 since the number of tests close to 100% (which is the points) in Figure 5 more than those in Figure 4.

Table 1: NIST Test Suite Results for the PRNG Based on Two Chaotic Maps [23]

PRNG Based on Two Chaotic Maps			
Test no.	Test name	p-value	Conclusion
1	Frequency	0.6936	PASSED
2	Block frequency	0.7740	PASSED
3	Runs	0.7489	PASSED
4	Longest run	0.1637	PASSED
5	Rank	0.7278	PASSED
6	FFT	0.6470	PASSED
7	Non-overlapping template	0.0401	PASSED
8	Overlapping template	0.1916	PASSED
9	Universal	0.3965	PASSED
10	Linear complexity	0.2187	PASSED
11	Serial (1)	0.1567	PASSED
12	Serial (2)	0.2624	PASSED
13	Approximate entropy	0.2101	PASSED
14	Cumulative sums (1)	0.4846	PASSED
15	Cumulative sums (2)	0.2366	PASSED
16	Random excursions	0.2938	PASSED
17	Random excursions variant	0.0633	PASSED

Table 2: NIST Test Suite Results for the QRWs-based PRNG [25]

QRWs-based PRNG			
Test no.	Test name	p-value	conclusion
1	Frequency	0.222465	PASSED
2	Block frequency (block = 128)	0.932368	PASSED
3	Runs	0.436267	PASSED
4	Longest run	0.388167	PASSED
5	Rank	0.436267	PASSED
6	Spectral DFT	0.180314	PASSED
7	Non-overlapping template (m = 9)	0.962460	PASSED
8	Overlapping template (m = 9)	0.577368	PASSED
9	Universal (block = 7)	0.596355	PASSED
10	Linear complexity (block = 500)	0.826735	PASSED
11	Serial (1) (block = 16)	0.719705	PASSED
12	Serial (2) (block = 16)	0.580439	PASSED
13	Approximate entropy (block = 10)	0.565844	PASSED
14	Cumulative sums (1)	0.438435	PASSED
15	Cumulative sums (2)	0.051895	PASSED
16	Random excursions (x = -1)	0.506488	PASSED
17	Random excursions variant (x = +1)	0.527057	PASSED

Table 3: NIST Test Suite Results for RC4

RC4			
Test no.	Test name	p-value	conclusion
1	Frequency	0.456	PASSED
2	Block frequency	0.658	PASSED
3	Runs	0.290	PASSED
4	Longest run	0.924	PASSED
5	Rank	0.514	PASSED
6	FFT	0.304	PASSED
7	Non-overlapping template	0.498	PASSED
8	Overlapping template	0.262	PASSED
9	Universal	0.596	PASSED
10	Linear complexity	0.367	PASSED
11	Serial	0.548	PASSED
12	Approximate entropy	0.983	PASSED
13	Cumulative sums	0.414	PASSED
14	Random excursions	0.483	PASSED
15	Random excursions variant	0.636	PASSED

Table 4: NIST Test Suite Results for Salsa20

Salsa20			
Test no.	Test name	p-value	conclusion
1	Frequency	0.494	PASSED
2	Block frequency	0.319	PASSED
3	Runs	0.182	PASSED
4	Longest run	0.304	PASSED
5	Rank	0.760	PASSED
6	FFT	0.052	PASSED
7	Non-overlapping template	0.511	PASSED
8	Overlapping template	0.740	PASSED
9	Universal	0.956	PASSED
10	Linear complexity	0.475	PASSED
11	Serial	0.212	PASSED
12	Approximate entropy	0.154	PASSED
13	Cumulative sums	0.421	PASSED
14	Random excursions	0.513	PASSED
15	Random excursions variant	0.526	PASSED

Table 5: NIST Test Suite Results for HC-128

HC-128			
Test no.	Test name	p-value	conclusion
1	Frequency	0.311	PASSED
2	Block frequency	0.310	PASSED
3	Runs	0.722	PASSED
4	Longest run	0.983	PASSED
5	Rank	0.910	PASSED
6	FFT	0.148	PASSED
7	Non-overlapping template	0.925	PASSED
8	Overlapping template	0.321	PASSED
9	Universal	0.370	PASSED
10	Linear complexity	0.569	PASSED
11	Serial	0.762	PASSED
12	Approximate entropy	0.768	PASSED
13	Cumulative sums	0.934	PASSED
14	Random excursions	0.297	PASSED
15	Random excursions variant	0.218	PASSED

Table 6: NIST Test Suite Results for SOSEMANUK

SOSEMANUK			
Test no.	Test name	p-value	conclusion
1	Frequency	0.679	PASSED
2	Block frequency	0.122	PASSED
3	Runs	0.276	PASSED
4	Longest run	0.384	PASSED
5	Rank	0.097	PASSED
6	FFT	0.081	PASSED
7	Non-overlapping template	0.494	PASSED
8	Overlapping template	0.319	PASSED
9	Universal	0.335	PASSED
10	Linear complexity	0.658	PASSED
11	Serial	0.927	PASSED
12	Approximate entropy	0.304	PASSED
13	Cumulative sums	0.477	PASSED
14	Random excursions	0.434	PASSED
15	Random excursions variant	0.464	PASSED

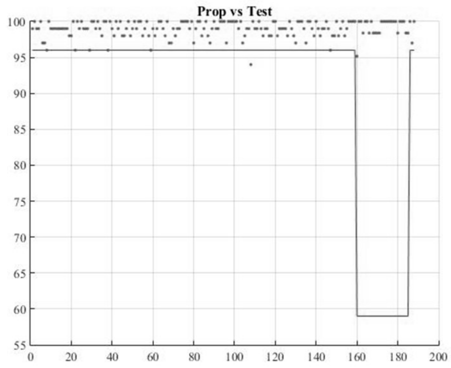

Figure 4. NIST Test Key Stream Results for Salsa20

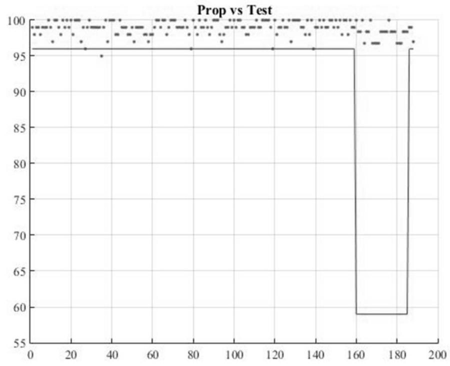

Figure 5. NIST Test Key Stream Results for RC4

4.2 Entropy Test Results

Table 7 represents the results obtained from applying the entropy test on the sets of data obtained from RC4, Salsa20 and the generator introduced in section 2.1.1. Since the output of Salsa20 is 64 bits, then we need to generate at least 2^{70} samples. However, it is very difficult to generate 2^{70} samples, as that would require huge amounts of storage. As a result, we divided the output of both Salsa20 and the chaos-based cipher into 8-bit blocks, and we applied the entropy test using the equation in section 3.2.

Table 7: Entropy Test Results

Generator	Entropy
RC4	7.99401
Salsa20 (Divided into 8-bit blocks)	7.99453
Chaos based cipher (Divided into 8-bit blocks)	7.99334

The optimal entropy value for 8-bit generator should be close to 8 and the percentage of RC4 is 0.99924 which is good result, while for Chaos based cipher is 0.9949 which is acceptable and less than RC4, for Salsa20 is 0.9952 which is also acceptable and less than RC4.

4.3 Hamming Distance Test Results

Table 8 represents the results obtained from applying the hamming distance test on the sets of data obtained from RC4, Salsa20 and the generator introduced in section 2.1.1. The steps used to obtain the results are as follows: we generated two sequences using each generator. Each sequence consists of 1,000,000 bits. However, the two keys used to generate each of the two sequences only differ in one bit. After we changed one bit in the input, we measured the amount of change between the two generated sequences using the equation defined in section 3.3. In this section we used a new methodology of calculating The HD, which is from research undergoing by master student [80]. This new methodology is based on the local and global HD, the global HD is the well-known HD test, while the local HD is the HD value per block or unit under the test, in the research of [80] is proved that some algorithms have HD values close to the optimal while the values is not random behavior. As an example assumes one block has 40% as HD value and the next block has 60% as HD value, it is clear that theses result are very bad while the global (normal HD value) is close to the optimal one. In our research, we use 8, 16 and 32 bits. The minimum local HD in Salsa20 is 25%, which means 16 bits are zeros and 48 are ones, while the maximum local HD is 76% which means 15.4 bits are ones and 48.6 bits are zeros, these maximum and minimum local HD values justifying the global HD value of Salsa20 which is close to the optimal (i.e. number of ones in the first local and the second local is almost equal to the number of zeros in the first and second locals), this is not a positive indicator of the uniformity distribution but also it gives an indication of acceptable security level. In RC4, the minimum local HD value is 0%, which means 8 bits are zeros and 0 are ones, while the maximum local HD is 100% which means 8 bits are ones and 0 bit is zeros, these maximum and minimum local HD values also justifying the global HD value of RC4 which is close to the optimal (i.e. number of ones in the first local and the second local is almost equal to the number of zeros in the first and second locals), the indicator in RC4 of the uniformity distribution is lower than Salsa20. In Chaos based cipher, the minimum local HD value is 9.38%, which means 3 bits are zeros and 29 bits are ones, while the maximum local HD is 81.25% which means 26 bits are ones and 6 bits are zeros, these maximum and minimum local HD values also justifying the global HD value of Chaos based cipher which is a little bit far from the optimal HD than Salsa20 and RC4. The indicator in Chaos based cipher of the uniformity distribution is lower than Salsa20. In order to verify those indicators, the three generators under the test (Salsa20, RC4 and Chaos based cipher) are reevaluated and the number of sequences having local HD more than 75% or less than 25% are calculated and presented in the same table. Our indicator regarding Salsa20 proves the robustness and uniformity distribution of the generated bits since the number of sequences are only one sequence. In RC4, also our indicator are true since the Percentage of local HD less than 25% is 0.034536 and almost the same for those more than 75% which are not negligible percentages. Regarding the Chaos-based cipher, the percentage can be negligible in some applications and not in other.

Table 8: HD Test Results

Generator	*Salsa20*	*RC4*	*Chaos-based cipher*
Global HD	50.0364%	50.0474 %	49.8969%
Min Local HD	*25%*	*0%*	*9.38%*
Max Local HD	*76.6%*	*100%*	*81.25%*
Percentage of Local HD less than 25%	*0*	*0.034536*	*0.001056*
Percentage of Local HD more than 75%	*0.000064*	*0.035192*	*0.001024*

4.4 Histogram and Chi-Square Test Results

Figures 6 and 7 represent the results obtained from applying the histogram test on the sets of data produced by the PRNGs reviewed in Section 2.1.1 and 2.1.2. While Figures 8 and 9 represent the results obtained from applying the histogram test on the set of data produced by RC4 and Salsa20 stream cipher respectively. Visually, it appears that the data is uniform. To confirm the obtained results the Chi-Square test is used. The theoretical value at P-value 0.05 is 293 and at P-value 0.1 is 287, which means the experimental values of the proposed generator lower than 293 are passed at P-value 0.05 and which are lower than 287 are passed the Chi-Square test at P-value 0.1. Table 9 presents the Chi-Square test for the three presented generator. It is clear, that RC4 and Salsa20 pass the test for both P-values, while the Chaos based cipher is passed the test at P-value 0.05 and failed at at P-value 0.1.

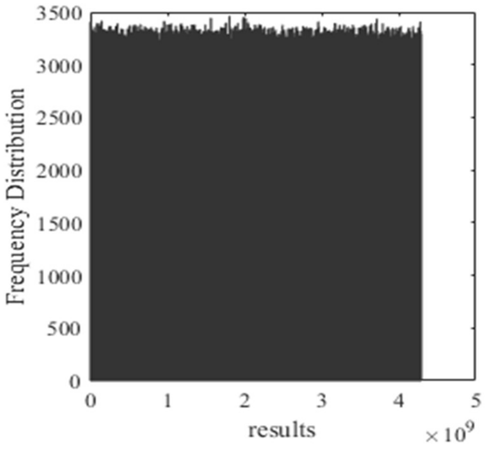

Figure 6. HISTROGRAM Test Results for Chaos-based stream cipher

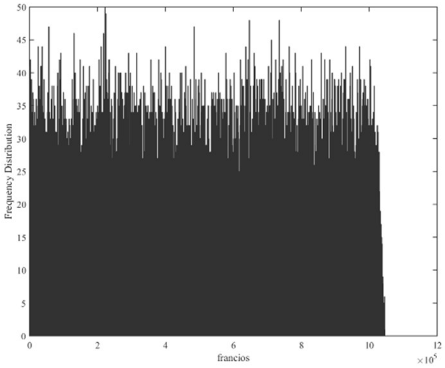

Figure 7. HISTROGRAM Test Results for PRNG Based on Two Chaotic Maps

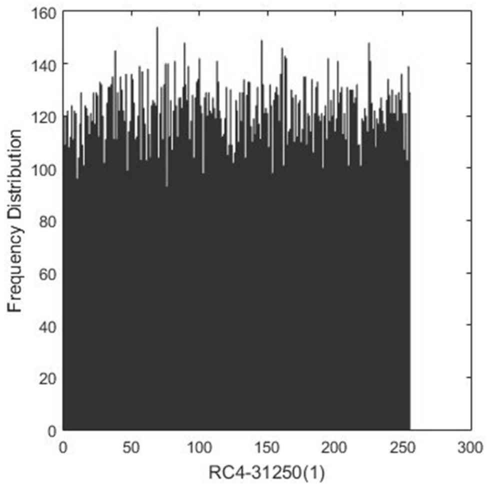

Figure 8. HISTOGRAM Test Results for RC4

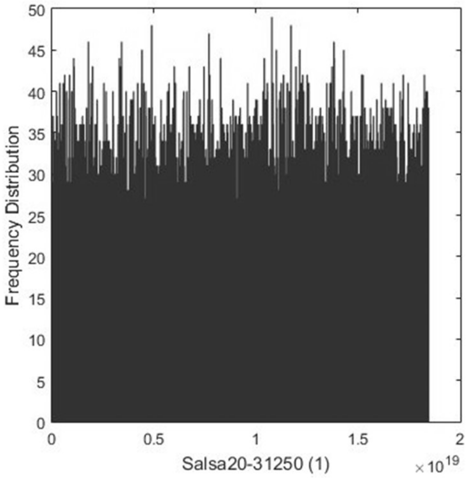

Figure 9. HISTOGRAM Test Results for Salsa20

Table 9: Chi-Square Test Results

PRNG/ Stream Cipher	Experimental value
RC4	259
Salsa20 (Divided into 8-bit blocks)	238
Chaos based cipher (Divided into 8-bit blocks)	289

4.5 Mapping Test Results

The mapping test point out of the dynamic behavior of the system. The obtained result in general with some exceptions, confirm the randomness of the proposed PRNGs and eSTREAM ciphers. Figures 10 and 11 represent the results obtained from applying the mapping test on the sets of data obtained produced by the

PRNGs reviewed in Section 2.1.1 and 2.1.2. While Figures 12 and 13 represent the results obtained from applying the mapping test on the set of data produced by RC4 and Salsa20 stream cipher respectively. The mapping result reflects the dynamic behavior of the system.

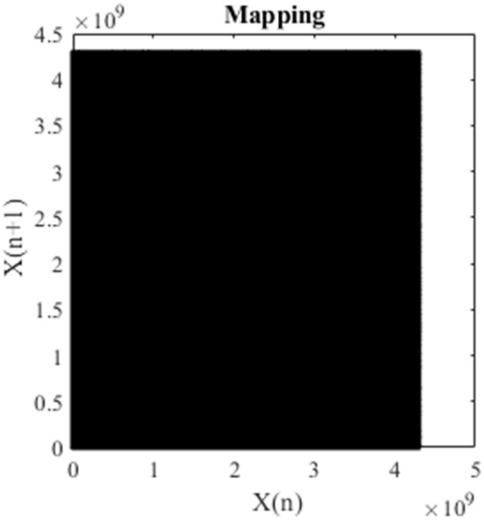

Figure 10. MAPPING Test Results for Chaos-based stream cipher

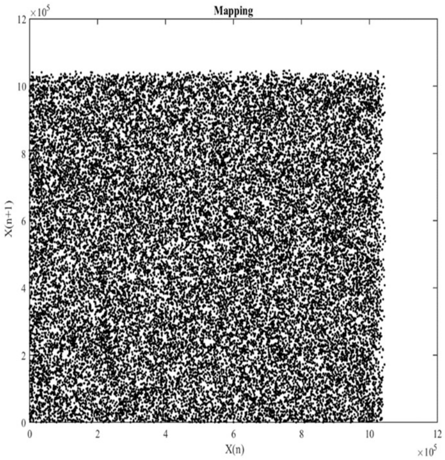

Figure 11. MAPPING Test Results for PRNG Based on Two Chaotic Maps

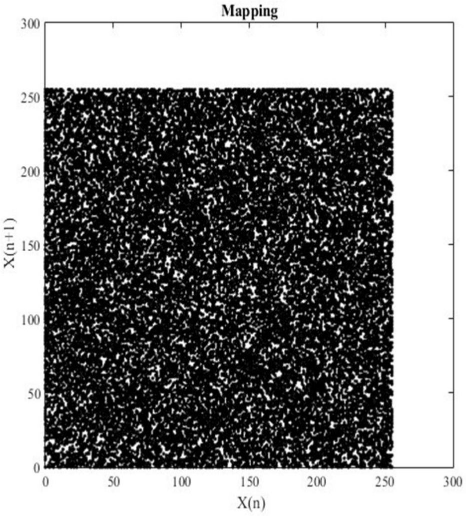

Figure 12. MAPPING Test Results for RC4

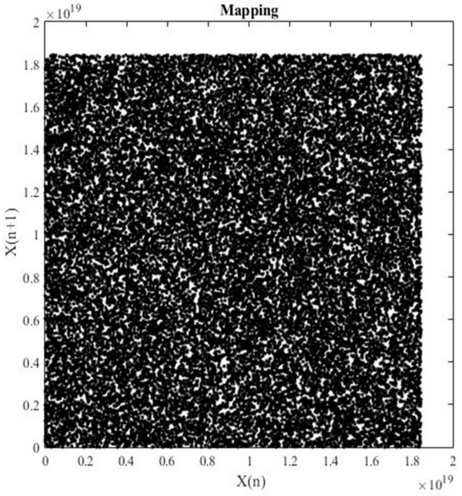

Figure 13. MAPPING Test Results for Salsa20

4.6 Correlation Test Results

Figures 14 and 15 show the correlation test on the set of data obtained from RC4 and Salsa20 stream ciphers respectively. The generated sequences are not correlated nor repeated.

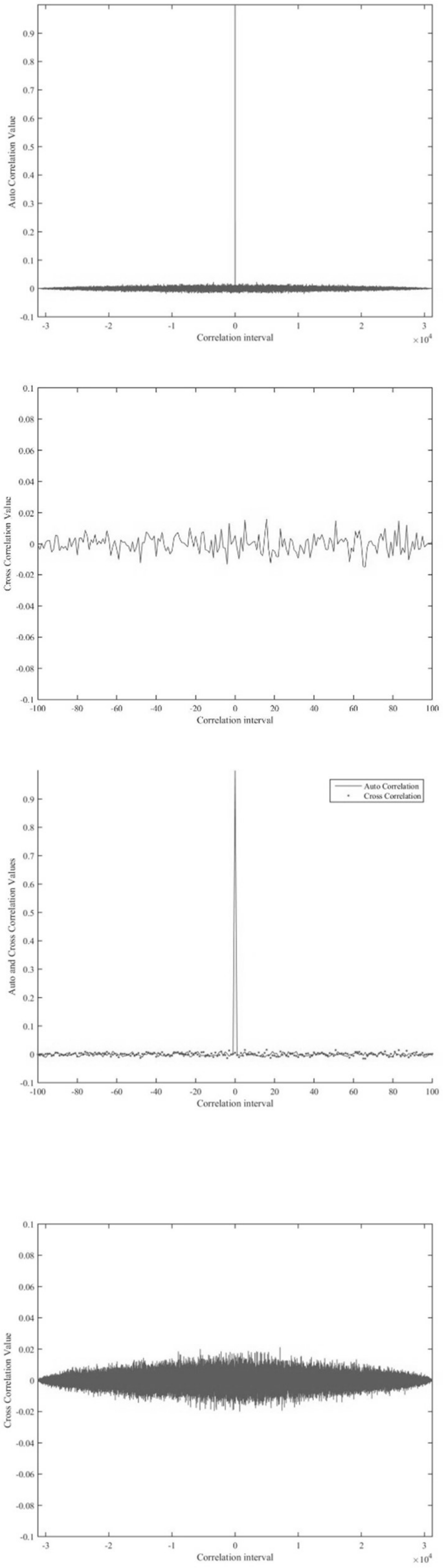

Figure 14. Correlation Test Results for RC4

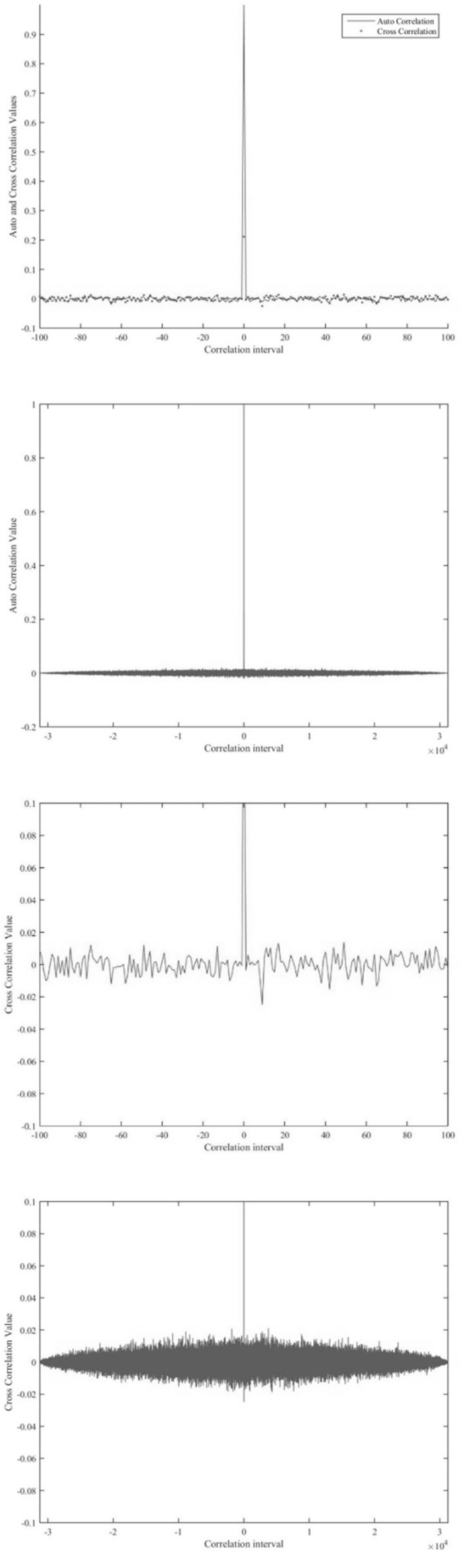

Figure 15. Correlation Test Results for Salsa20

RC4 has passed approximate entropy test with the highest p-value compared with the other generators, as shown in table 3. It also passed the longest run test with a very high p-value. However, it passed the runs test but with a relatively small p-value. While Salsa20 passed some tests with very high p-values, including the overlapping template test and the serial test, it passed other test with relatively small p-values. As shown in table 4, Salsa20 has passed the runs test and the FFT test with p-values of 0.182 and 0.052 respectively. These values are the lowest compared to other tested generators.

The HC-128 stream cipher has passed many tests with the highest p-value. It has the highest p-value for the runs test, the longest run test, the rank test and the cumulative sums test. It also passed other tests with a very good p-value. However, it passed a few tests with smaller p-values, including the FFT test and the frequency test, as shown in table 5.

The results obtained from SOSEMANUK were average as shown in table 6. None of the p-values were relatively high. Moreover, it passed some tests with the lowest p-value such as the rank test and the block frequency test.

4.7 Time and Performance Test Results

In this section we analyzed the performances of different stream cipher models. For each stream cipher algorithm, we measured the time of encryption/ decryption in (µs), Bitrate in (MBit/s) and Number of Cycles to generate one Byte [68] (NCpB). (see Equations 12 and 13).

$$\mathrm{BR} = \frac{Data\ Size_{(Mbit)}}{GT_{(\mu s)}} \qquad (12)$$

$$\mathrm{NCpB} = \frac{CPU\ Speed_{(Hertz)}}{BR_{(Mbit\,/\,s)}} \qquad (13)$$

Tables 10 and 11 show the computed performance results for the implemented stream cipher algorithms. We applied this comparison over a set of data produced by each model. Each set of data has an equal size of 3MBs. The results provided in Tables 10 and 11 indicate that the eSTREAM project ciphers have very good results in term of computing performance. The NCpB is between 9 to 14 cycles in encryption /decryption using eSTREAM project ciphers, while it is too high using the RC4. The NCpB for the RC4 cipher is approximately four times of eSTREAM ciphers' NCpB. This result reflects the admirable performances that the eSTREAM ciphers have over the standard RC4 stream cipher.

Table 10: Time and Performance Results for Encryption Operation

Encryption					
Model	**Size (B)**	**Time for 1000 Encrypt. (us)**	**Time (ns / B)**	**Bit Rate (Mbps)**	**Number of Cycle for 1 Byte (Cycles / B)**
RC4	3145728	26946708	8.60	533.24	56.3
Salsa20	3145728	13483978	4.29	1866.35	9.9
HC-128	3145728	19647606	6.25	1280.86	14.4
SOSEMANUK	3145728	14134923	4.49	1780.40	10.4

Table 11: Time and Performance Results for Decryption Operation

Decryption					
Model	**Size (B)**	**Time for 1000 Decrypt. (us)**	**Time (ns / B)**	**Byte Rate (Mbps)**	**Number of Cycle for 1 Byte (Cycles / B)**
RC4	3145728	29843025	10.20	448.81	55.1
Salsa20	3145728	16020572	5.09	1570.84	11.7

HC-128	3145728	22671643	7.21	1110.01	16.7
SOSEMANUK	3145728	16841003	5.35	1494.32	12.4

5. CONCLUSION

In this survey, some of the states of the art researches that were proposed in the field of PRNG are highlighted. We started by a short introduction in the vast domain of randomness and types of random number generators. Then, we reviewed standard stream ciphers and some of the researches that were proposed in the field of PRNG. We gave an introduction to encryption and reviewed the two main types of encryption. The process of testing random numbers using statistical and visualization tests was discussed in the following section. We gave a description for NIST test suite, visualization test, histogram test, chi-square test, correlation test and mapping test. As an example, we presented the results obtained from the application of some statistical tests over sets of data obtained from various ciphers, such as Salsa20 and RC4. In section 4.3, a new methodology of Hamming Distance is presented which is proved that some algorithms and generators can pass the HD while the local HD is not good random behavior. All tested generators successfully passed the NIST test suite with good p-value numbers. The results of applying histogram, correlation and mapping tests were also presented in various figures. Finally, we compared between various generators in terms of encryption and decryption speed. The results of this comparison were summarized in two tables.

We have seen that some of the previously used generators fail severely in the histogram test. Does that mean that we shouldn't use these generators at all? Well, it depends on the target application. For example, in cryptology, we require generators which are unpredictable in a specific sense. Such generators should pass all statistical tests, but their current limitation is that they are not fast enough for real-time applications. Research is still under way. It's also worth noting that we faced some challenges whilst performing some of the statistical tests. One of the limitations of current technology is the storage. Some tests require generating huge amounts of data when applied to 32-bit generators or higher. This data cannot be stored on any hard disk drive. As a solution, we divided the output of such generators into 8-bit chunks that require significantly less storage. It is hoped that the reader has developed an appreciation of this subject and has recognized the importance of testing generators using various tests and tools.

6. FUTURE WORK

Based on our study and analysis of PRNGs, the need of a generator that passes all statistical tests in both local and global is required. We are looking to design and implement a new PRNG that passes all local and global statistical tests with high throughput.

REFERENCES:

[1]. Schulz, Marc-André, et al. "Analysing humanly generated random number sequences: a pattern-based approach." *PloS one* 7.7 (2012): e41531

[2]. Elminaam DS, Abdual-Kader HM, Hadhoud MM. Evaluating the performance of symmetric encryption algorithms. IJ Network Security. 2010 May 1;10(3):216-22.

[3]. Chen G, Mao Y, Chui CK. A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos, Solitons & Fractals. 2004 Jul 31;21(3):749-61.

[4]. Zhu Sh. Algorithm Design Of Secure Data Message Transmission Based On Openssl And Vpn. Journal Of Theoretical & Applied Information Technology. 2013 Feb 10;48(1).

[5]. Bellare M, Rogaway P. Optimal asymmetric encryption. In Workshop on the Theory and Application of of Cryptographic Techniques 1994 May 9 (pp. 92-111). Springer, Berlin, Heidelberg.

[6]. Simmons GJ. Symmetric and asymmetric encryption. ACM Computing Surveys (CSUR). 1979 Dec 1;11(4):305-30.

[7]. Elminaam DS, Abdual-Kader HM, Hadhoud MM. Evaluating the performance of symmetric encryption algorithms. IJ

Network Security. 2010 May 1;10(3):216-22.

[8]. Agrawal M, Mishra P. A comparative survey on symmetric key encryption techniques. International Journal on Computer Science and Engineering. 2012 May 1;4(5):877.

[9]. Blumenthal, Uri, Fabio Maino, and Keith McCloghrie. *The advanced encryption standard (AES) cipher algorithm in the SNMP user-based security model*. No. RFC 3826. 2004.

[10]. Zeng K, Yang CH, Wei DY, Rao TR. Pseudorandom bit generators in stream-cipher cryptography. Computer. 1991 Feb;24(2):8-17.

[11]. Stipčević M, Koç ÇK. True random number generators. In Open Problems in Mathematics and Computational Science 2014 (pp. 275-315). Springer International Publishing.

[12]. Haahr, Mads. "Introduction to randomness and random numbers." *Random. org, June* (1999).]

[13]. Hamidouche, Wassim, et al. "Selective video encryption using chaotic system in the SHVC extension." Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015.

[14]. O'toole JE, Tuttle JR, Tuttle ME, Lowrey T, Devereaux KM, Pax GE, Higgins BP, Ovard DK, Yu SS, Rotzoll RR, inventors; Micron Technology, Inc., assignee. Pseudo random number generator. United States patent US 6,314,440. 2001 Nov 6.

[15]. Kelsey J, Schneier B, Wagner D, Hall C. Cryptanalytic attacks on pseudorandom number generators. In Fast Software Encryption 1998 (pp. 168-188). Springer Berlin/Heidelberg.

[16]. James F. A review of pseudorandom number generators. Computer Physics Communications. 1990 Oct 1;60(3):329-44.

[17]. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-Allen and Hamilton Inc Mclean Va; 2001 May 15.

[18]. Gaeini, Ahmad, Abdolrasoul Mirghadri, and Gholamreza Jandaghi. "A General Evaluation Pattern for Pseudo Random Number Generators." *Trends in Applied Sciences Research*10.5 (2015): 231

[19]. Taha, Mohammed Abu, et al. "Design and efficient implementation of a chaos-based stream cipher." International Journal of Internet Technology and Secured Transactions 7.2 (2017): 89-114.

[20]. Francois M, Grosges T, Barchiesi D, Erra R. A new pseudo-random number generator based on two chaotic maps. Informatica. 2013 Jan 1;24(2):181-97.

[21]. Protopopescu, Vladimir A., Robert T. Santoro, and Johnny S. Tolliver. "Fast and secure encryption-decryption method based on chaotic dynamics." U.S. Patent No. 5,479,513. 26 Dec. 1995.].

[22]. Savvidy KG. The MIXMAX random number generator. Computer Physics Communications. 2015 Nov 30;196:161-165

[23]. Yang YG, Zhao QQ. Novel pseudo-random number generator based on quantum random walks. Scientific reports. 2016;6

[24]. Akhshani, A., et al. "Pseudo random number generator based on quantum chaotic map." *Communications in Nonlinear Science and Numerical Simulation* 19.1 (2014): 101-111

[25]. Rivest R. RSA Security response to weaknesses in key scheduling algorithm of RC4. Technical note, RSA Data Security, Inc. 2001 Aug.

[26]. Stallings W. The RC4 Stream Encryption Algorithm. Cryptography and network security. 2005.

[27]. Chowdhury, Dipanwita Roy, Vincent Rijmen, and Abhijit Das. "Progress in Cryptology-INDOCRYPT 2008." *9th international conference on cryptology in India, Kharagpur, India*. 2008.

[28]. Maitra S, Paul G. Analysis of RC4 and Proposal of Additional Layers for Better Security Margin. In INDOCRYPT 2008 Dec 14 (Vol. 5365, pp. 27-39).

[29]. Roos A. A class of weak keys in the RC4 stream cipher.

[30]. Biham E, Granboulan L, Nguyên PQ. Impossible Fault Analysis of RC4 and Differential Fault Analysis of RC4. In FSE 2005 Jan 1 (Vol. 2005, pp. 359-367).

[31]. Mister S, Tavares SE. Cryptanalysis of RC4-like Ciphers. In Selected areas in cryptography 1998 Aug 17 (Vol. 1556, pp. 131-143).

[32]. AlFardan NJ, Bernstein DJ, Paterson KG, Poettering B, Schuldt JC. On the Security of RC4 in TLS. In USENIX Security Symposium 2013 Aug 14 (pp. 305-320).

[33]. Paul S, Preneel B. A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. In Fast Software Encryption 2004 (pp. 245-259). Springer Berlin/Heidelberg.

[34]. Klein A. Attacks on the RC4 stream cipher. Designs, Codes and Cryptography. 2008 Sep 1;48(3):269-86.

[35]. Golic JD. Linear statistical weakness of alleged RC4 keystream generator. InEurocrypt 1997 May 11 (Vol. 1233, pp. 226-238).

[36]. Fluhrer SR, McGrew DA. Statistical analysis of the alleged RC4 keystream generator. In FSE 2000 Apr 10 (Vol. 1978, pp. 19-30).

[37]. Mantin I, Shamir A. A practical attack on broadcast RC4. In International Workshop on Fast Software Encryption 2001 Apr 2 (pp. 152-164). Springer, Berlin, Heidelberg.

[38]. Fluhrer S, Mantin I, Shamir A. Weaknesses in the key scheduling algorithm of RC4. In Selected areas in cryptography 2001 Aug 16 (Vol. 2259, pp. 1-24).

[39]. Maitra S, Paul G. Analysis of RC4 and Proposal of Additional Layers for Better Security Margin. In INDOCRYPT 2008 Dec 14 (Vol. 5365, pp. 27-39).

[40]. Robshaw M, Billet O, editors. New stream cipher designs: the eSTREAM finalists. Springer; 2008 Jun 19.

[41]. Bernstein DJ. Salsa20 design.

[42]. Bernstein DJ. Salsa20 specification. eSTREAM Project algorithm description.

[43]. Bernstein DJ. ChaCha, a variant of Salsa20. In Workshop Record of SASC 2008 Jan (Vol. 8, pp. 3-5).

[44]. Bernstein DJ. Salsa20/8 and Salsa20/12. eSTREAM, ECRYPT Stream Cipher Project, Report. 2006;7.

[45]. Tsunoo Y, Saito T, Kubo H, Suzaki T, Nakashima H. Differential cryptanalysis of Salsa20/8. InWorkshop Record of SASC 2007 (p. 12).

[46]. Crowley P. Truncated differential cryptanalysis of five rounds of Salsa20. The State of the Art of Stream Ciphers SASC. 2006 Feb 2;2006:198-202.

[47]. Fischer S, Meier W, Berbain C, Biasse JF, Robshaw MJ. Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. In Indocrypt 2006 Nov 27 (Vol. 4329, pp. 2-16).

[48]. Shi Z, Zhang B, Feng D, Wu W. Improved key recovery attacks on reduced-round salsa20 and chacha. In International Conference on Information Security and Cryptology 2012 Nov 28 (pp. 337-351). Springer, Berlin, Heidelberg.

[49]. Bernstein DJ. Extending the Salsa20 nonce. In Workshop record of Symmetric Key Encryption Workshop 2011 Feb (Vol. 2011).

[50]. Gierlichs B, Batina L, Clavier C, Eisenbarth T, Gouget A, Handschuh H, Kasper T, Lemke-Rust K, Mangard S, Moradi A, Oswald E. Susceptibility of eSTREAM candidates towards side channel analysis.

[51]. Wu H. The stream cipher HC-128. Lecture Notes in Computer Science. 2008 Jan 1;4986:39-47.

[52]. Salehani YE, Kircanski A, Youssef A. Differential fault analysis of sosemanuk. In International Conference on Cryptology in Africa 2011 Jul 5 (pp. 316-331). Springer, Berlin, Heidelberg.

[53]. Wu, Hongjun. "A new stream cipher HC-256." *International Workshop on Fast Software Encryption*. Springer, Berlin, Heidelberg, 2004

[54]. Kircanski A, Youssef AM. Differential Fault Analysis of HC-128. In Africacrypt 2010 May 3 (pp. 261-278).

[55]. Maitra S, Paul G, Raizada S, Seb S, Sengupta R. Some observations on HC-128. Design, codes and Cryptography. 2011 Apr 1 ; 59(1):231-45

[56]. Sekar G, Preneel B. Improved Distinguishing Attacks on HC-256. IWSEC. 2009 Oct 5;5824:38-52.

[57]. Paul G, Maitra S, Raizada S. A Theoretical Analysis of the Structure of HC-128. In International Workshop on Security 2011 Nov 8 (pp. 161-177). Springer, Berlin, Heidelberg.

[58]. Gierlichs B, Batina L, Clavier C, Eisenbarth T, Gouget A, Handschuh H, Kasper T, Lemke-Rust K, Mangard S, Moradi A, Oswald E. Susceptibility of eSTREAM candidates towards side channel analysis.

[59]. Hell M, Johansson T, Brynielsson L. An overview of distinguishing attacks on stream ciphers. Cryptography and Communications. 2009 Apr 1;1(1):71-94.

[60]. Banegas G. Attacks in Stream Ciphers: A Survey. IACR Cryptology ePrint Archive. 2014 Aug 26;2014:677.

[61]. Berbain C, Billet O, Canteaut A, Courtois N, Gilbert H, Goubin L, Gouget A, Granboulan L, Lauradoux C, Minier M, Pornin T. Sosemanuk, a fast software-oriented stream cipher. eSTREAM report 2005/027 (2005). URL: http://www. ecrypt. eu. org/stream/papers. html. Citations in this document.;3.

[62]. Salehani YE, Kircanski A, Youssef A. Differential fault analysis of sosemanuk. In International Conference on Cryptology in Africa 2011 Jul 5 (pp. 316-331). Springer, Berlin, Heidelberg.

[63]. Feng X, Liu J, Zhou Z, Wu C, Feng D. A Byte-Based Guess and Determine Attack on SOSEMANUK. InASIACRYPT 2010 Dec 2 (pp. 146-157).

[64]. Ahmadi H, Eghlidos T, Khazaei S. Improved guess and determine attack on SOSEMANUK. ECRYPT Stream Cipher Project, Report. 2005;85:2005.

[65]. Lee JK, Lee DH, Park S. Cryptanalysis of Sosemanuk and SNOW 2.0 Using Linear Masks. In ASIACRYPT 2008 Dec 7 (Vol. 5350, pp. 524-538).

[66]. Lin D, Jie G. Guess and determine attack on sosemanuk. In Information Assurance and Security, 2009. IAS'09. Fifth International Conference on 2009 Aug 18 (Vol. 1, pp. 658-661). IEEE.

[67]. Ma Z, Gu D. Improved differential fault analysis of SOSEMANUK. In Computational Intelligence and Security (CIS), 2012 Eighth International Conference on 2012 Nov 17 (pp. 487-491). IEEE.

[68]. Leander G, Zenner E, Hawkes P. Cache Timing Analysis of LFSR-Based Stream Ciphers. In IMA Int. Conf. 2009 Dec 2 (pp. 433-445).

[69]. Lee, Jung-Keun, Dong Hoon Lee, and Sangwoo Park. "Cryptanalysis of SOSEMANUK and SNOW 2.0 using linear masks." *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Berlin, Heidelberg, 2008

[70]. Chen H, Wang T, Guo S, Zhao X, Zhang F, Liu J. Improved Differential Fault Analysis of SOSEMANUK with Algebraic Techniques. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2017 Mar 1;100(3):811-21.

[71]. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-Allen and Hamilton Inc Mclean Va; 2001 May 15.

[72]. Shannon CE. A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review. 2001 Jan 1;5(1):3-55.

[73]. http://vikimy.com/l-en/Entropy_(information_theory) , last visited 4.3.2018

[74]. Farajallah M, El Assad S, Chetto M. Dynamic adjustment of the chaos-based security in real-time energy harvesting

sensors. In Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing 2013 Aug 20 (pp. 282-289). IEEE.

[75]. Jolfaei A, Mirghadri A. An image encryption approach using chaos and stream cipher. Journal of Theoretical and Applied Information Technology. 2010 Sep;19(2):117-25.

[76]. Li, Ming, Zhongxian Peng, and Hai Nan. "A modified reversible data hiding in encrypted images using random diffusion and accurate prediction." *Etri Journal* 36.2 (2014): 325-328.

[77]. Farajallah, Mousa, et al. "ROI encryption for the HEVC coded video contents." Image Processing (ICIP), 2015 IEEE International Conference on. IEEE, 2015.

[78]. Farajallah, Mousa. Chaos-based crypto and joint crypto-compression systems for images and videos. Diss. UNIVERSITE DE NANTES, 2015.

[79]. Rukhin, Andrew, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-Allen and Hamilton Inc Mclean Va, 2001.

[80]. Zaher Amro, Master of Informatics, Palestine Polytechnic University, Derive a standard mathematical tools to evaluate Image encryption algorithms, 2016, supervised by Dr. Mousa Farajallah.