Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

# PRNG based Symmetric Stream Cipher

**J K M Sadique Uz Zaman**
Institute of Radio Physics and Electronics, University of Calcutta,
92, Acharya Prafulla Chandra Road, Kolkata - 700 009, India
Email: jkmsadique@gmail.com

----------ABSTRACT----------

**An algorithm of symmetric stream cipher is proposed embedding a PRNG within it. Stream ciphers require a random key stream. A simple exclusive OR operation is needed between the randomly generated key stream and the text character to obtain the cipher text. Randomness of the cipher text depends on the randomness of the key stream. A comparative study on randomness of three key streams namely RC4, PM and BBS is observed in this paper. In the study, 13424 key-bits are generated in each key sequence. For each algorithm 300 such key sequences are generated using different keys. The randomness characteristics of 300 key sequences of each algorithm are tested using four basic statistical tests. Results are compared and it is observed that using a PRNG in the proposed algorithm makes it statistically secured.**

Keywords - **PRNG, BBS, Statistical Test, Stream Cipher, RC4.**

Date of Submission: 13, February 2013 Date of Acceptance: 11, March 2013

## 1. INTRODUCTION

In the field of symmetric stream cipher RC4 is the most popular algorithm that is immune to attack [1]. It used variable key size for ciphering with byte-oriented operation. A number of research activities have been done to analyze the RC4 algorithm [2-8]. Some of these papers mentioned the weaknesses present in the algorithm [2-4]. Recent trend is towards the designing of hardware architecture on RC4 [9-11]. Claude Shannon proved a necessary condition for a symmetric key ciphering scheme to be unconditionally secure is that $H(K) \geq H(M)$, where H() denotes the entropy function while random variables K and M respectively denoting the secrete key and plaintext message [12]. In cryptography the one-time pad is an unconditionally secure algorithm, but it has a drawback that key should be as long as the message. The ciphering that uses pseudorandom algorithm does not offer unconditional security, but it may provide computational security. In the field of electronic communication, encryption of message is very necessary to hide the meaning of information from unauthorized user. And the objective of encryption process is to make the message randomized so that it would be difficult to find out the meaning of the message. In symmetric stream ciphers generally the exclusive XOR operation is performed between the plain text and a key generated stream. The strength of a cipher depends on how much random key stream is produced by the key scheduling algorithm. More random key stream would produce more random cipher by encryption. There is many Pseudo Random Bit Generator (PRBG) and Pseudo Random Number Generator (PRNG) in literatures. It is well known that Blum Blum Shub (BBS) generator is a cryptographically secured PRBG to provide pseudorandom bit stream [1,13]. But it is considered as a slower algorithm as it produce only single bit at a time. To make faster execution of the BBS generator it is proved that $r \leq \log_2\log_2(n)$ number of bits can be extracted at a time where n is the modulus used in the algorithm [14].

Motivation of this work is to find a simple but robust symmetric stream cipher algorithm using a PRNG or a PRBG. Implementation of this proposed algorithm is very easy and assumed that it would be used in wide range of application. In this present algorithm r is considered as 3 to achieve more efficiency than extracting only a single bit. The value of r may be increased but it would be complicated to calculate in a 32 bit processor. Here the BBS generator is used in two different ways that would be discussed later in this paper. Another process of PRNG named by Park-and-Miller algorithm is also considered in this paper [1,15]. To choose the better one between the two algorithms BBS and Park-and-Miller four basic statistical tests have been performed on the key streams obtained by incorporating the two algorithms. The tests are described in chapter 5 of [12, 16]. A brief description of the pseudorandom algorithms used here is given in Section 2. The processes of encryption adopted in this paper are discussed in Section 3. Information on the statistical tests and results obtained by the tests are mentioned respectively in Sections 4 and 5. And the conclusion of this paper is pointed out in Section 6.

## 2. PSEUDO RANDOM ALGORITHM

In reality there is no algorithm to produce pure random numbers. All the algorithms are deterministic in nature. And the output sequences produced by them are not random, rather they are called pseudorandom. There are different types of generators like Linear Congruential generator, Quadratic Congruential generator etc. A Quadratic as well as a Linear Congruential generator is discussed respectively in Section 2.1 and 2.2.

Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

### 2.1. Pseudo Random Bit Generator (PRBG)

The BBS algorithm is a popular and well known pseudorandom bit generator. In cryptographic algorithms, it has perhaps the highest security. Brief information of this generator along with its security is as follows:

Let p and q are two large prime numbers and both are congruent to 3 modulo 4. That is

$$p \equiv q \equiv 3 \pmod 4 \quad (1)$$

this means that for both p and q the remainder is 3 when divided by 4.

$$\text{Let} \quad n = p \times q \quad (2)$$

Again let s is a randomly chosen number relatively prime to n, that is, neither p nor q is a factor of s. Then the BBS generator will produce a stream of bits $b_i$ using the following algorithm:

$X_0 = s^2 \bmod n$
for i = 1 to ∞
  $X_i = (X_{i-1})^2 \bmod n$
  $b_i = X_i \bmod 2$

In each iteration the LSB is considered as a random bit.

It is widely believed that there is no polynomial time Monte Carlo algorithm for Composite Quadratic Residues with small error probability [14]. The BBS generator follows the Composite Quadratic Residue property. And this provides some evidence that BBS generator has provable security against cryptographic attack.

It is to be observed that the computational approach of this algorithm is very simple and it does not need more variables. Hence it is easy to implement in software as well as in hardware. The only drawback with this algorithm is that it's a slow on. If $r \le \log_2\log_2(n)$ number of bits are extracted in each iteration and $r \ge 2$ then the generator will be faster. Now let one intends to extract 4 bits, that is, r = 4 then $n \ge 2^{2^r} = 2^{16} = 65536$. But it can be proved from criteria (1) and (2) that as p and q both are odd, n must be an odd number, means n > 65536 and the $X_i \le n - 1$. In this way if any time $X_i \ge 65536$, then $(X_i)^2$ will not fit in 32-bit processor. Similarly if r = 3, then $n \ge 256$. And there is a wide range for n as, $256 \le n \le 65535$. For this reason r = 3, p = 131 and q = 499 are taken in this proposed algorithm. Statistical results obtained from extracting 1 and 3 bits each time are given in Table 3 respectively on BBSG-1 and BBSG-3 rows.

### 2.2. Pseudo Random Number Generator (PRNG)

The Park-and-Miller algorithm [15] that has been chosen to study the effect of incorporating randomness in ciphering is

$$\text{seed} = A \times \text{seed} \,\%\, M \quad (3)$$

where M is a large prime integer and A should also be a large integer less than M. The value of M is chosen as the maximum positive number in signed long integer mode, which is (231 – 1) = 2147483647, also a prime number. The value of A is taken as $A < \sqrt{M}$. In the present testing purpose A is considered as 16807.

While multiplying A with seed, the product may not fit into the 32-bit computer memory and the algorithm may fail due to the storage problem. A technique, presented following Schrage [17,18] to handle the memory overflow problem, is as follow:

Q = M / A;
R = M % A;
M = A × Q + R and R < min(A, Q).
Also, k = seed / Q;
r = seed % Q;
seed = Q × k + r;

In order that R < min(A, Q), A should be an integer as close to the √M. Now,

A×seed = A×(Q×k + r)
= A×Q×k + A×r
= k×(M – R) + A×r
= k×M + (A×r – k×R), if (A×r – k×R) ≥ 0
= (k–1)×M +{M +(A×r – k×R)}, if (A×r – k×R) < 0

The programming technique to bypass the storage problem is coded in a PM block below:

PM block:
Q = M / A;
R = M % A;
k = seed / Q;
r = seed % Q;
if ((A×r – k×R) ≥ 0) $\text{seed}_{j+1}$ = (A×r – k×R);
otherwise, $\text{seed}_{j+1}$ = M + (A×r – k×R);

The PM algorithm generates non-repeating integral seeds between 0 and M, if the initial seed is non-zero positive integer less than M. For 0 or M, the algorithm fails. In order that 0 or M is accepted as an initial seed, XOR operation between seed and an arbitrary integer (MASK) is introduced before and after the PM block [19]. The code then looks like,

seed = seed^MASK;

PM block
seed = seed^MASK;

The above algorithm may also fail, if MASK happens to be a successive seed. Such a situation can be overcome, if the statement involving the first XOR operation is changed and rewritten as,

if(seed != MASK) seed = seed^MASK;

PM block
seed = seed^MASK;

Considering the innovation proposed in the coding of the Park-and-Miller algorithm, the symbolic code narrated above is sound and shall accept any seed. The Park-and-Miller algorithm is a reasonably good PRNG for cryptographic applications though the period of randomness in it is not of very high order. In this paper modulo operation with modulus 256 is applied on seed in equation (3) to get 8-bit key stream, i.e., the operation given in equation (3.1) is added after equation (3).

$$\text{key} = \text{seed} \,\%\, 256 \quad (3.1)$$

Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

## 3. ENCRYPTION PROCESSES

Incorporating a PRBG or a PRNG in symmetric stream cipher algorithm was the primary issue of this study. It was considered that the algorithm used only a Secured Personal Number instead of key. Four types of algorithms are considered as (i) BBS1: used BBSG algorithm and 1-bit is extracted each time, (ii) BBS3: used BBSG algorithm and 3-bits are extracted each time, (iii) PM: used Park-and-Miller algorithm and (iv) RC4: the original RC4 algorithm available in [1]. These four algorithms are discussed here in Sections 3.1 through 3.3.

### 3.1. Encryption with a PRBG

The encryption technique using BBS algorithm is thought in two different ways. In one type of technique only one bit is extracted each time while in another technique three bits are extracted each time. The two different processes namely BBS1 and BBS3 are discussed respectively in Sections 3.1.1 and 3.1.2.

#### 3.1.1. Encryption with BBS1

This technique, named as BBS1 in this paper, incorporates BBS algorithm which is discussed in section 2.1. In this algorithm, only one bit is extracted in a single iteration and concatenated to the right of the key stream. Hence to generate required 13424 bits the BBS algorithm will execute for $t_1$ = 13424 times. The key stream is used in ciphering from left to right.

#### 3.1.2. Encryption without BBS3

The technique is named as BBS3 in this paper. Its procedure is almost similar to the BBS1. The only different is that, it extracts three bits and concatenated to the key stream. The first bit extracted is the LSB and it is concatenated first. Then the second and third bits are concatenated in their respective order. To generate 13424 bits the BBS algorithm will execute $t_3$ = 13424/3 times. If $t_3$ is real number then its ceiling value will be taken to produce the required number of bits.

### 3.2. Encryption with a PRNG

The encryption process can also be done using a PRNG. Here the Park-n-Miller PRNG presented in Section 2.2 is innovatively introduced in the proposed algorithm known as PM. Working process of the algorithm is as follows:

A key stream is generated using successive operations of equations (3) and (3.1) as mentioned in Section 2.2. In each iteration a 8-bit key is generated. Hence to get 13424 bits only 1678 iterations are needed. The key stream is used sequentially to produce cipher text.

### 3.3. RC4 algorithm

In the field of stream cipher, RC4 is widely used and very simple algorithm. It is byte-oriented variable key-size algorithm designed by Ron Rivest in 1987 for RSA Data Security [1]. Though its design procedure is very simple, yet it is very strong against cryptographic attacks. The RC4 was kept as trade secret till 1994 when it was leaked out [4]. Today RC4 is a part of many network protocols, e.g. SSL/TLS, WEP, WPA and many others. There were many cryptanalysis to look into its key weaknesses [3, 4] followed by many new stream ciphers. RC4 is still the popular stream cipher since it is executed fast and provides high security.

Design procedure of 256-byte state vector S which is used as the key-pool in RC4 is very simple. The entries in S is initialized as S[0]=0, S[1]=1, …, S[255]=255. Let K is the given key of length *keylen* A temporary vector T of 256-byte is created from given key of length as T[i] = K[i **mod** *keylen*]. Next T is used to initial permutation of S. The process is as follow:

```
j = 0;
for i = 0 to 255 do
  j = (j + S[i] + T[i]) mod 256;
  Swap (S[i], S[j]);
```

After the permutation of S vector, the input key is longer used. This S vector is used to provide a sequence of key stream k as follow:

```
i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```

## 4. STATISTICAL TESTS

The Standard normal distribution or Gaussian distribution and the $\chi^2$ (chi-square) distribution both are the two widely used distributions in statistical applications. Before entering into the statistical tests a preliminary knowledge of these two distributions are essential. To explain and understand these distributions few related definitions are described following the chapter 5 of [12].

**Definition 1:** Let $X$ is the result of an experiment. Then $X$ is said to be continuous random variable if $X \in$ R, where R is the set of real numbers.

**Definition 2:** A probability density function of a continuous random variable $X$ is a function $f(x)$ satisfying the criteria:

(i) $f(x) \geq 0$ for all $x \in$ R;

(ii) $\int_{-\infty}^{\infty} f(x)\,dx = 1$; and

(iii) Probability $Pr(a < X \leq b) = \int_a^b f(x)\,dx$ for $a, b \in$ R.

**The normal distribution:** If a large number of independent random variables having same *mean* ($\mu$) and *variance* ($\sigma^2$) are summed up then it follows the normal distribution and can be denoted as $N(\mu, \sigma^2)$.

**Definition 3:** A continuous random variable $X$ is in $N(\mu, \sigma^2)$ if its probability density function is defined as,

$$f(x) = \exp[-(x-\mu)^2/2\sigma^2] / [\sigma\sqrt{(2\pi)}], \text{ where } -\infty<x<\infty \quad (4)$$

Any $X$ is said to have standard normal distribution if $X$ is in $N(0, 1)$, that is $\mu = 0$ and $\sigma^2 = 1$. A typical graph of $N(0, 1)$ distribution is shown in Fig.1.

Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

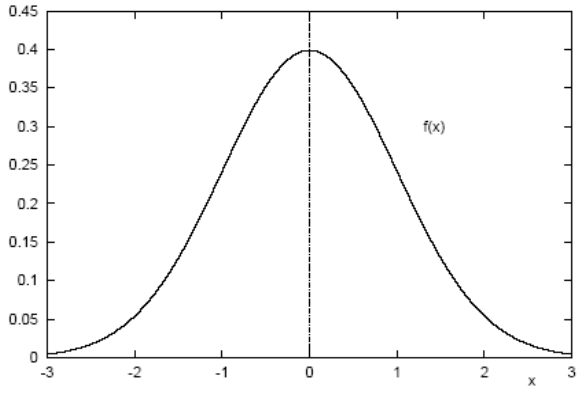


Figure 1. The standard normal distribution $N(0, 1)$

**The chi-square ($\chi^2$) distribution:** If the squares of $\upsilon$ independent random variables having standard normal distribution are summed up then it follows the $\chi^2$ distribution with $\upsilon$ degrees of freedom. The $\chi^2$ distribution is very important and is used to compare the calculated frequencies of events to their expected frequencies under a hypothetical distribution.

**The gamma function $\Gamma(t)$:** Gamma function is defined as

$$\Gamma(t) = {}_0\int^{\infty} x^{t-1}e^{-x}\,dx, \quad \text{for } t > 0. \tag{5}$$

**Definition 5:** Let degrees of freedom $\upsilon \geq 1$ be an integer. Then a continuous random variable $X$ has a $\chi^2$ distribution with $\upsilon$ degrees of freedom if its probability density function is defined as,

$$f(x) = \begin{cases} x^{(\nu/2)-1}\, e^{-x/2} / \ \Gamma(\nu/2)2^{\nu/2}, & \text{for } 0 \leq x < \infty \\ 0, & \text{for } x < 0 \end{cases} \tag{6}$$

In this distribution $\mu = \upsilon$ and $\sigma^2 = 2\upsilon$. A typical graph of $\chi^2$ distribution with $\upsilon = 7$ is shown in Fig.2.

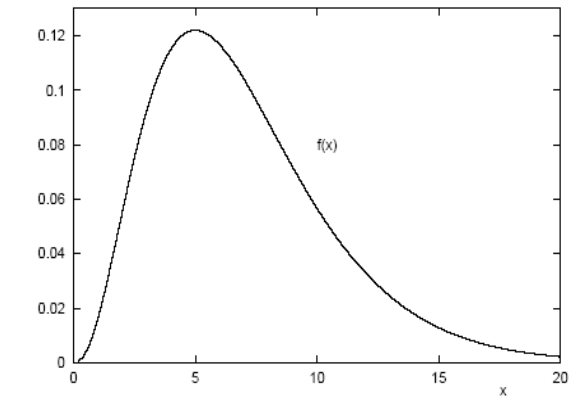


Figure 2. The $\chi^2$ distribution with degrees of freedom $\upsilon = 7$

To check the randomness of PRBG or PRNG algorithm, the following four statistical tests are executed on the key streams generated by the algorithm. The length of the key stream (n) used for testing purpose was set to 13424 bits.

**Frequency test:** Through this test it is intended to see if the frequencies of 0s and 1s across the entire n-bit sequence are approximately equal, as would be expected for a random sequence. If the number of 0s and 1s are not same, it is intended to see if their difference falls within the limit of randomness. Let $n_0$ and $n_1$ denotes the number of 0s and 1s respectively in the sequence and $n = n_0 + n_1$. If $n \geq 10$, then statistic used is,

$$X_1 = (n_0 - n_1)^2 / n \tag{7}$$

which approximately follows $\chi^2$ distribution with $\upsilon = 1$.

**Serial test (two-bit test):** In n-bit random sequence ($n \geq 21$), every 2-bit pattern has the same chance of occurrence as would be expected for a random sequence. Let $n_0$, $n_1$, $n_{00}$, $n_{01}$, $n_{10}$ and $n_{11}$ denotes the number of 0, 1, 00, 01, 10 and 11 respectively in a sequence. The statistic used is,

$$X_2 = [4(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) / (n-1)] - [2(n_0^2 + n_1^2) / n] + 1 \tag{8}$$

which approximately follows $\chi^2$ distribution with $\upsilon = 2$.

**Poker test:** In long n-bit random sequence every m-bit pattern has the same chance of occurrence as would be expected of a random sequence where $k = \text{floor}(n/m) \geq 5(2^m)$. Let $n_i$ be the number of occurrences of the $i^{th}$ type of sequence of m length, $1 \leq i \leq 2^m$, then the statistic used is,

$$X_3 = (2^m/k)\ ({}_{i=1}\textstyle\sum^{2m} n_i^2) - k \tag{9}$$

which approximately follows $\chi^2$ distribution with $\upsilon = 2^m - 1$. If $m = 1$, then it is similar to the frequency test.

**Runs test:** Runs of length k means exactly k identical bits bounded by bits of opposite value. A run of 0s is called a gap and a run of 1s is called a block. In this test it is intended to see if the frequencies of gaps and blocks of various lengths are in limits of randomness. In a random sequence of length n the expected number of gaps (or blocks) of length i is $e_i = (n - i + 3) / 2^{i+2}$. Let k = largest integer i for which $e_i \geq 5$. Also let $B_i$, $G_i$ be the number of blocks and gaps respectively of length i for each $1 \leq i \leq k$. The statistic used is,

$$X_4 = {}_{i=1}\textstyle\sum^{k} [(B_i - e_i)^2/e_i] + {}_{i=1}\textstyle\sum^{k} [(G_i - e_i)^2/e_i] \tag{10}$$

which approximately follows $\chi^2$ distribution with $\upsilon = 2k - 2$.

## 5. RESULTS AND DISCUSSIONS

The four statistical test-algorithm described in Section 4 are used for statistical randomness testing of incorporating a PRBG and PRNG in proposed algorithm. The

Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

significance level $\alpha = x$ means a test should be considered as non random if percentage of test results are failed at least $100x\%$ of times. Therefore $\alpha = 0.05$ indicates that a test should be non random if its result fails at least 5% times. In Table 1, test wise threshold values for $\alpha = 0.05$ are given. A bit-sequence is said to be passed a test if its $X_i$ value does not greater than the corresponding threshold value. For example, a bit-sequence will be considered as passing the Frequency test if $X_1 \leq 3.8415$.

Table 1: Threshold values of the statistical tests for $\alpha$=0.05 and n = 13424

| | **Frequency test ($X_1$)** | **Serial test ($X_2$)** | **Poker test ($X_3$)** | **Runs test ($X_4$)** |
|---|---|---|---|---|
| Degrees of freedom ($\upsilon$) | 1 | 2 | 255 | 16 |
| Threshold | 3.8415 | 5.9915 | 293.2478 | 26.2962 |

To take a good decision on acceptance or rejection of any algorithm a large number of sample sizes must be considered. Considering n = 13424, $\alpha = 0.05$ and sample size = 300, a typical results of $X_i$ values corresponding to the four basic tests are shown in Table 2. From Table 2, it is observed that, PM has passed the Frequency test 96% times while RC4 has passed the same test 94.333% times.

Table 2: Percentage of passing the statistical tests

| **Data set** | **Frequency test** | **Serial test** | **Poker test** | **Runs test** | **All (4) tests** |
|---|---|---|---|---|---|
| RC4 | 94.333 | 93.667 | 94.667 | 92.667 | 81.333 |
| PM | 96.000 | 95.667 | 97.333 | 95.000 | 88.000 |
| BBS1 | 22.333 | 4.333 | 0 | 0 | 0 |
| BBS3 | 50.333 | 35.000 | 0 | 0 | 0 |

In PM algorithm, M is a large prime which is considered 2147483647 in this paper. The next seed may be any one within this value. To get a random key character, modulo operation with modulus 256 is executed. That is same key character may occur from any one of M/256 seeds. Hence if one knows a key character, yet it would be difficult to find the next key character. The difficulty depends on M, larger the M difficulty will be high.

## 6. CONCLUSION

From Table 2, it is very easy to draw a conclusion that Park-and-Miller (PM) algorithm is satisfying all the four tests, where every test has passed at least 95% of times. The RC4 is considered as a base algorithm to test other algorithms like PM and BBS in this paper and it is also evident that PM has passed in comparison to the RC4 too. On the other side it is clear that BBS1 and BBS3 both show very unsatisfactory results, though BBS is known as cryptographically secured one. Most probably the reason for failure of BBS is that, in this paper very small values of n in equation (2) is considered, where BBS is assumed to have large value of it. A further study is required to test the BBS algorithm with large n, and hope that with large n BBS also provide a very good security like RC4 and PM. Finally it can be concluded that, the use of Park-and-Miller algorithm in symmetric stream ciphering technique would provide statistically secured cipher text.

## 7. ACKNOWLEDGEMENTS

I express my sincere gratitude towards the UGC, New Delhi for providing financial support. I am also indeed thankful to my research guide Prof. Ranjan Ghosh for his support and inspiration. Last but not least, I express my heartiest gratitude to the Institute of Radio Physics and Electronics, University of Calcutta for providing necessary facilities towards research.

## REFERENCES

[1] W. Stallings, *Cryptography and Network Security* (Delhi, Pearson Education, 4th Edition, 2008).

[2] S. Maitra, G. Paul, Analysis of RC4 and proposal of additional layers for better security margin, *Proc. Indocrypt*, 2008, IIT Kharagpur, LNCS 5365, 27-39.

[3] S. Paul, B. Preneel, A New Weakness in the RC4 Keystream. Generator and an Approach to Improve the. Security of the Cipher, *Proc. Fast Software Encryption*, 2004, Berlin, LNCS 3017, 245-259.

[4] S. Fluhrer, I. Mantin, A. Shamir, Weakness in the Key Scheduling Algorithm of RC4, *Proc. Int. Workshop on Selected Areas in Cryptography*, 2001, Toronto, LNCS 2259, 1-24.

[5] I. Mantin, A. Shamir, A Practical Attack on Broadcast RC4, *Proc. Fast Software Encryption*, 2001, Japan, LNCS 2355, 152-164.

[6] S. Fluhrer, D. McGrew, Statistical Analysis of the Alleged RC4 Key Stream Generator, *Proc. Fast Software Encryption*, 2000, New York, LNCS 1978, 19-30.

[7] L. Knudsen, et al., Analysis Method for Alleged RC4, *Proc. ASIACRYPT*, 1998, Beijing, LNCS 1514, 327-341.

[8] S. Mister, S. Tavares, Cryptanalysis of RC4-Like Ciphers, *Proc. Int. Workshop on Selected Areas in Cryptography*, 1998, Canada,LNCS 1556, 131-143.

[9] S.S. Gupta, K. Sinha, S. Maitra, B.P. Sinha, One Byte per Clock: A Novel RC4 Hardware, *Proc. Indocrypt*, 2010, Hyderabad, LNCS 6498, 347-363.

Volume: 04 Issue: 05 Pages: 1725-1730 (2013) ISSN : 0975-0290

[10] P. Kitsos, G. Kostopoulos, N. Sklavos, O. Koufopavlou, Hardware Implementation of the RC4 stream Cipher, *Proc. 46th IEEE Midwest Symposium on Circuits & Systems*, 2003, Cairo, Vol.3, 1363-1366.

[11] D.P. Matthews, Jr. System and method for a fast hardware implementation of RC4, US Patent No. 6549622, Campbell, CA, April 2003.

[12] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography* (CRC Press, 1996) 169-190.

[13] L. Blum, M. Blum, M. Shub, A Simple Unpredictable Pseudo-Random Number Generator, *SIAM Journal on Computing*, 15(2), 1986, 364-383.

[14] D.R. Stinson, *Cryptography Theory and Practice* (Boca Raton, Chapman & Hall, CRC, 3rd Edition, 2006).

[15] S. K. Park, K. W. Miller, Random Number Generators: Good ones are hard to find, *Communications of the ACM*, 31(10), 1988, 1192 – 1201.

[16] www.cacr.math.uwaterloo.ca/hac

[17] L. Schrage, A More Portable Fortran Random Number Generator, *ACM Transactions on Mathematical Software*, 5(2), 1979, 132-138.

[18] P. Bratley, B.L. Fox, L.E. Schrage, *A Guide to Simulation*, (New York, Springer-Verlag, 1983).

[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (New York, Cambridge University Press, 2nd Edition, 1988) 274-328.

**AUTHOR'S BIOGRAPHY**

*J K M Sadique Uz Zaman* received his M.Sc. and MCA degree from Tilka Manjhi Bhagalpur University respectively in 2002 and 2006. He is pursuing his Ph.D. (Tech) at the University of Calcutta. Presently he is doing research as a UGC selected SRF in Engineering and Technology. He has 1 review paper in an International Journal and 4 papers in National Conferences/Workshops. He has qualified the GATE-2012 in Computer Science and Information Technology and also qualified the UGC-NET December 2012 for Lectureship in Computer Science and Applications.