International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 6 (2017), pp. 1461-1470

http://www.ripublication.com

Comparative Analysis of SLA-LFSR with Traditional Pseudo Random Number Generators

Mohammed Abdul Samad AL-khatib
Jamia Hamdard, Hamdard Nagar, New Delhi-110062, India.

Auqib Hamid Lone
Independent Security Researcher, Srinagar Jammu and Kashmir 192221, India.

Prof. Moin Uddin
Jamia Hamdard, Hamdard Nagar, New Delhi-110062, India.

Abstract

This paper does performance studies and analysis of SLA-LFSR (Secure Lightweight Acoustic) PRNG with traditional PRNGs (BBS, LCG,). Traditional PRNGs are first generation PRNGs, which are very popular since their inception while SLA-LFSR is new but overshadows traditional PRNGs due to its immunity against a wide variety of threats and attacks and thus considered as cryptographically secure PRNG. The performance of traditional PRNGs is degraded by the factors like initialization time and reseeding time. The study and analysis suggest that SLA-LFSR is the optimal PRNG among all compared PRNGSs sir to its long periodic length, optimal memory, and CPU usage along with improved security.

I. INTRODUCTION

A Random Number Generator is an algorithm which is utilized to produce an irregular, unpredictable sequence of numbers. Producing random numbers is an essential requirement of cryptography, random numbers are not only used in generating cryptographic keys, but are also required in security protocols and in the steps of

cryptographic algorithms they also needed in other areas such as statistic, game theory, simulation, and most important the cryptography.

There is two types of random number generators namely true random number generator (TRNG) and pseudo-random number generator (PRNG). TRGN is hardware that plugs into a computer and generates actual random numbers using the specially prepared physical process to extract the entropy from the natural phenomena. The major problem associated with TRNG is its expensive cost, and it does not produce random numbers at the required rate by many applications, while the PRNG is a mathematical algorithm that produces sequence of numbers which are not truly random but exhibit properties of random numbers.

PRNG takes initial value called seed as an input and expands it to produce a large quantity of pseudo-random numbers depending on the deterministic algorithm being used. The overall security of PRNG lays on its seed and the internal state, if anyone of them is compromised the entire sequences will become predictable. PRNG overcomes the cost and speed problem of TRNG. A PRNG that is required for cryptosystems is called a Cryptographically Secure Pseudorandom Number Generator (CSPRNG), which is based on the fundamental that the adversary should not be able to predict further output if they are unaware of the seed.

Requirements of CSPRNG:

- It should pass statistical randomness tests.
- It should be stable under critical attacks, even the initial state or current state been leaked.
- It should pass the next bit test. It means using the previous outputs; the attacker should not be able to predict the next upcoming output.
- It should be infeasible to predict the future state of CSPRNG by injecting known entropy.
- It should be ridden out of state compromise extensions attack.

In this paper, we have compared different performance and security aspects of BBS, LCG, LFSR and SLA-LFSR.

II. BLUM BLUM SHUB

Lenore Blum, Manuel Blum, and Michael Shub proposed in 1986 a Blum Blum Shub (BBS) generator [1] which was one of most popular pseudo-random number generators. Its security is based on the problem of factoring the Blum integers in polynomial time as factoring the Blum integers is a hard problem, and BBS usually generates imbalance

sequences having equal frequency of 0's and 1's and passed all the statistical properties [2] [3], and its properties are detailed [4].

Blum integers is a positive integer N which is a product of two distinct primes p and q, where $p \equiv q \equiv 3\ (mod\ 4)$. The p and q should be large primes because they decide the linear complexity of producing sequences which is $\pi/2$, and the period of maximum length which is π. Where $\pi = 2p_2q_2$.

BBS generator is a deterministic algorithm which represented as $x^2\ mod\ N$ and produces pseudorandom bits of size $log_2(log_2\ N)$

Algorithm of BBS generator:

1. $N \leftarrow pq$ where p and q are distinct prime and both are congruent to 3 (mod 4).
2. $x_0 \leftarrow s$ Where s is random integer in the interval $[1\ , N-1]$ and $gcd(s, N) = 1$
3. **For** $i \leftarrow 1$ to π
4. $x_i \leftarrow x_{i-1}^2\ mod\ N$
5. **OUTPUT:** $x_i\ \ mod\ 2$
6. **LOOP END**
7. **GOTO** step 2

Advantages of BBS:

- These sequences will have a maximum possible period.
- The sequences produced by the $x^2\ mod\ P$ generator are usually not balanced.

Disadvantages of BBS:

- This generator is very slow
- There is a proof lessening its security to the computational complexity of resolving the quadratic residue problem.
- Take more time for reseeding compared to other generators.
- Its maximum length period is less compared to LFSR and thus required frequent reseeding.

III. LINEAR CONGRUENTIAL GENERATOR

D. H. Lehmer introduced a linear congruential generator in 1949 [5] which is a deterministic algorithm that generates a sequence of numbers using the following linear recurrence, these sequences are not truly random but exhibit many properties of random numbers.

$x_i = (a\, x_{i-1} + c)mod\, m$

x_0, a, c, m are the parameters of linear congruential generator where

m is the modulus $m > 0$

a is the multiplier $0 < a < m$

c is the increment $0 \leq c < m$

x_0 is the seed $0 \leq x_0 < m$

LCG passes the three tests proposed by Stephen and Keith [6] for evaluating a random number generator, but it suffers from short period length which depends on the values of LCG parameters. If m is prime and $c = 0$ then for certain values of a the period of LCG is $m - 1$. It is highly recommended the value of m should be equal to the power of 2, then if c is relatively prime to m and $a = 1 + 4\, k$ where k is an integer the possible largest period will be equal to m, and if $c = 0$, x_0 is odd and a is given by $a = 3 + 8\, k$ or $a = 5 + 8\, k$ the possible largest period will be equal to $m/4$.

Algorithm of LCG:

1. Select a, c, m
2. Select seed x_0
3. **For** $i \leftarrow 1$ to $m - 1$
4. $x_i \leftarrow (a\, x_{i-1} + c)\, mod\, m$
5. **OUTPUT:** x_i
6. **LOOP END**
7. **GOTO** step 2

Advantages of LCG:

- LCG is quick and requires negligible memory.
- LCG is easy to implement on both the hardware and software.

Disadvantages of LCG:

- LCG produces poor quality sequences, as the generated numbers x_i and x_{i+1}

are dependent thus the sequence is predictable, by knowing a portion of the sequence the attacker can constructs the whole sequence.

- LCG could not be used for cryptography purpose or where there is a demand for high-quality randomness.
- The output of LCG is a classical cipher known as an affine cipher which is vulnerable to frequency analysis attack.
- Lower order bits of LCG have a shorter period as compared to the whole sequence.

IV. LINEAR FEEDBACK SHIFT REGISTER

A linear feedback shift register (LFSR) is an n-bit shift register which combines some bits of it states in exclusive-OR to form a feedback, the position of bits that are selected to form the feedback is known as taps. LFSR is an appropriate choice to simulate truly random binary sequences and have the maximum length period of $2^n - 1$ [7]. The LFSR is completely specified by its polynomial, and the period length is depends on the taps position, to achieve a period length of $2^n - 1$ the taps should be even and the set of taps should be relatively prime.

Advantages of LFSR:

- LFSRs have extended periodic length; a 32-bit LFSR has a periodic length greater than four billion.
- Fast and required minimal computation.
- Easy to implement in software or hardware.

Disadvantages of LFSR:

- LFSR cannot be directly used to produce random numbers; it required a method that combines the output of LFSR in an appropriate way to generate random numbers.
- LFSR itself is not secure as the output streams of LFSR are deterministic, knowing the current state of LFSR and the taps position make the next state predictable.

V. SLA-LFSR

SLA-LFSR [8] is a hybrid random number generator that extracts the entropy by sampling the audio, and it is based on the secure use of LFSR. SLA-LFSR has a periodic length of $2^{32} - 1$ and it does not require complex computation to initialize or

reseed the generator, it also passed all the randomness tests contains in NIST statistical test suite. The SLA-LFSR uses different cryptographic primitive to immunize the generator against the well-known attacks on PRNGs. Along with the security the SLA-FSR also provides high-speed generation of high-quality random numbers, and it is also compatible with CUDA implementation where it adopts the massively parallel platform of GPU to enhance its speed.

SAL-LFSR is prone to the following attacks:

- **Direct cryptanalytic attack:** Where the assailant tries to get the knowledge about the internal state or foresee the future output of PRNG by knowing about past output.
- **Input-Based attacks:** Where the aggressor gains the control of PRNG's data sources and becomes ready to alter the input of PRNG which permits them to predict the inner state of the PRNG.
- **Chosen input attack:** In this sort of attack, the aggressor can straightforwardly control the input of PRNG, to constrain the generator to cycle or re-hash a particular past output.
- **Replayed input attack:** where the attacker replays the current input without altering it.
- **Known input attack:** where the aggressor utilizes the knowledge of the input to constrain the conceivable output of PRNG.
- **State Compromise Extension Attacks:** the assailant utilizes the knowledge of compromised state to infer the past or future output.
- **Correlation-Attack:** It is the most popular assault on LFSR based PRNG; it misuse a statistical weakness of PRNGs.

Advantages of SLA-LFSR:

- Have excellent protection against the major attacks on PRNG.
- The high production rate of high-quality pseudorandom numbers.
- Suitable to implement on GPU or the cluster of GPUs to increase its generation rate to millions of high-quality pseudorandom numbers per seconds.

VI. RESULT AND ANALYSIS

The NIST has introduced the statistical test suite [9] which consist of 15 tests to check whether the generated sequence of bits is random or not. These tests do not assure that

the sequence is truly random instead of that they check whether the occurrence of bits is independent or not which is represented by p-value and it should be between 0.01 and 1.00. The above generators were subjected to these tests and all of them passed all the tests except the linear congruential generator which only passed the following tests: serial test, frequency test, golomb's randomness postulates, poker test, autocorrelation test, runs test, and Maurer's universal statistical test. Table1 shows the result of comparative analysis between above generators on the basis of memory and CPU requirement, randomness, period length, the time consumed during initialization, security and random number generating speed. To compare the speed of pseudorandom number generator, we used a system with the following comparison Intel i3-4030U with two cores clocked at 1.90 GHz and 4 GB DDR3 random access memory with the frequency of 1600Hz. In speed comparison, the major delay in the performance of BBS was due to frequent reseeding, and it consumed a large amount of time to compute an appropriate seed. If we let the BBS generator to completes its production without reseeding and allowing the reputation of same sequences it will take 839 milliseconds to produce three million numbers. The LFSR used in this comparison is of 32-bit with a period length greater than four billion. LCG and LFSR are fast, but they are not secure, SLA-LFSR has balanced between the security and speed. Initialization time is the time required to seed the generator first time and the time needed to reseed the generator after completing its cycle. Period length is the quantity of random numbers that the generator produces without repeating the generated sequence. Figure 3 shows the speed comparison between the selected PRNGs. Figure 4,5,6,7 show the memory and CPU capacity requirement to generate three million random numbers or bits for BBS, LCG, LFSR, and SLA-LFSR respectively the time unit in these figures are in seconds. Figure 1 shows the graphical representation of speed comparison between the selected PRNGs. It is clearly shown in Figure 3 that the SLA-LFSR is faster than BBS because the SLA-LFSR have not be reseeded due to its extended period but the BBS was reseeded more than thousand times for generating three million random numbers, and the BBS generator required a considerable amount of time and CPU capacity for calculating the new seed. The speed of LFSR is greater than the speed of LCG the delay here is also due to reseeding while LCG does not pass the minimum-security requirements and have short periodic length compared to LFSR.

Figure 2 shows the memory and CPU usage of BBS generator which consumes 18.1 MB of RAM and 27% of CPU and there is no fluctuation in the CPU usage and required 75 seconds to generate three million random numbers. Figure 3 shows the resource consumption by LCG which is 18.1 MB of main memory, 25% of CPU and 7.5 seconds to complete its execution. Figure 4 shows the resources requirement of LSFR which is 24.2 MB of RAM, 25% of CPU and 4.68 seconds. Figure 5 shows the resources usage by SLA-LFSR which is 32 MB of main memory, 75% of CPU and 45 seconds. There is fluctuation in CPU usage by SLA-LFSR which is due to using multiple threads to complete its task.

Table 1: Comparison analysis of PRNGs

	BBS	LCG	LFSR	SLA-LFSR
Initialization/reseeding time	Too large	Large	Less	Less
Memory usage	18.1 MB	18.1MB	24.2MB	32 MB
CPU usage	27%	25%	25%	75%
Period length	Short ($\pi = 2p_2q_2$)	Too short ($m-1$ at most)	Very long ($2^n - 1$)	Very long ($2^{32} - 1$)
Randomness	Passed all the statistical tests	Passed few statistical tests	Passed all the statistical tests	Passed all the statistical tests
Security	Normal	Not secure	Low	High
Speed (Time required to generate 3 million random numbers)	68000 ms	7620 ms	4680 ms	45010 ms

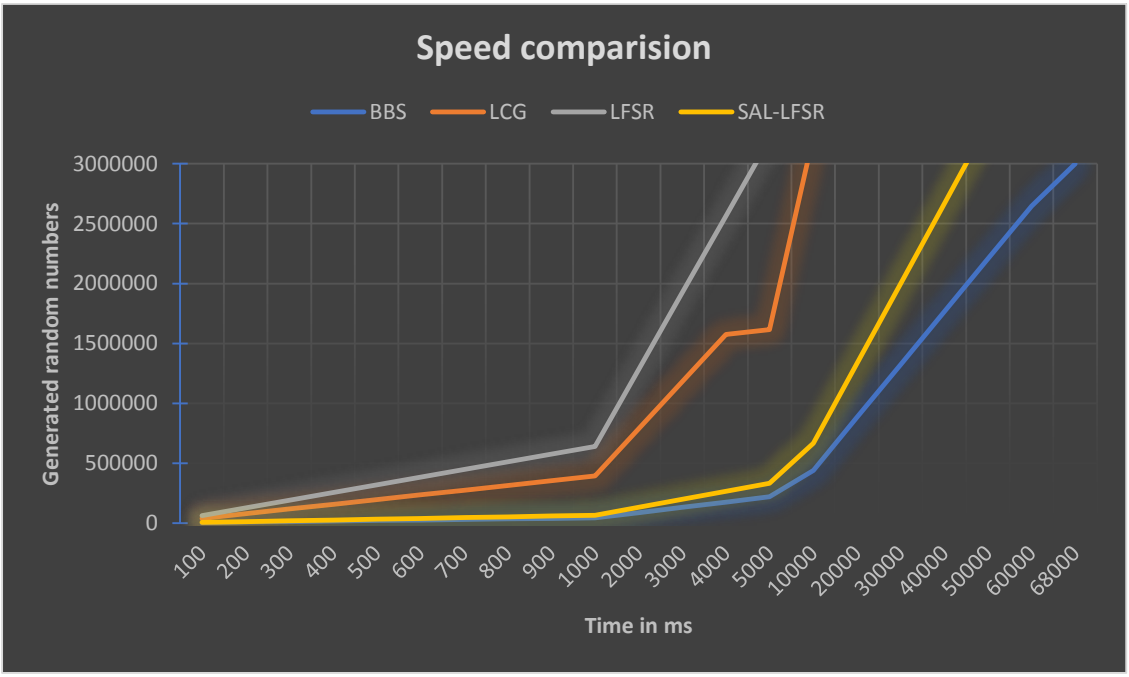

Figure 1: Graphical representation of speed comparison

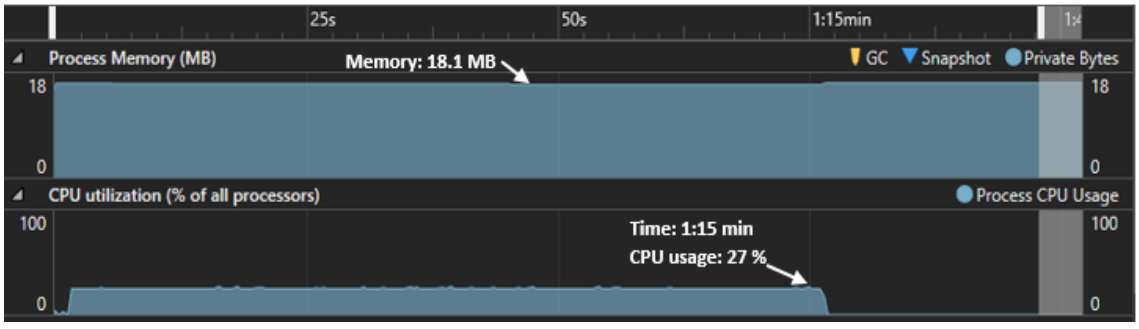

Figure 2: CPU and memory usage of BBS

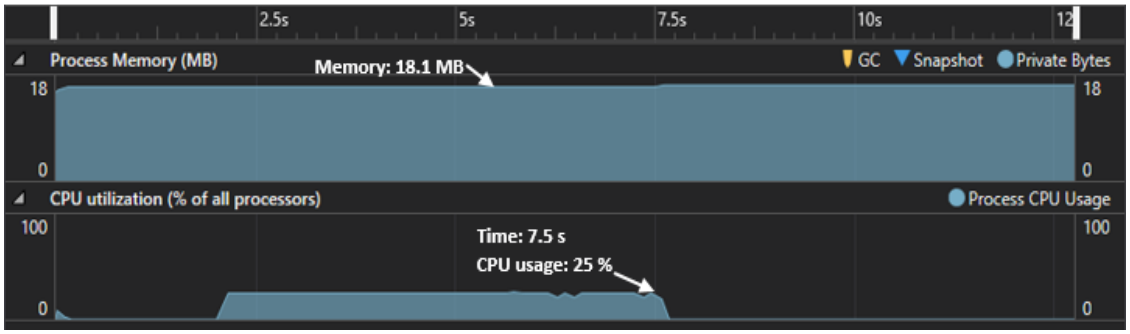

Figure 3: CPU and memory usage of LCG

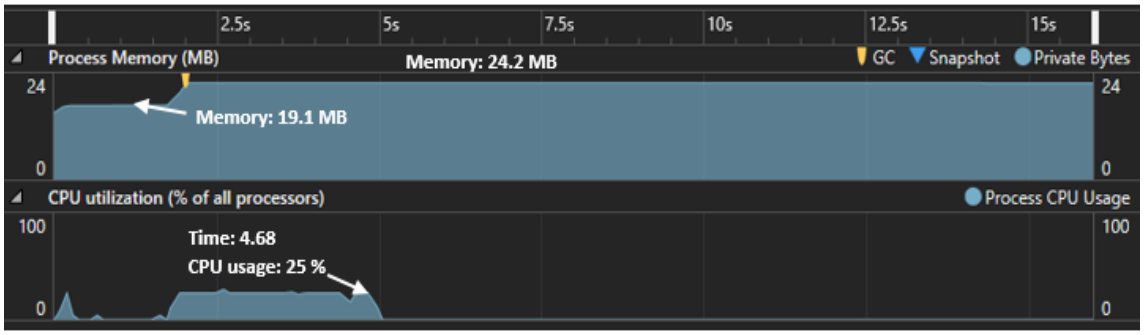

Figure 4: CPU and memory usage of 32-bit LFSR

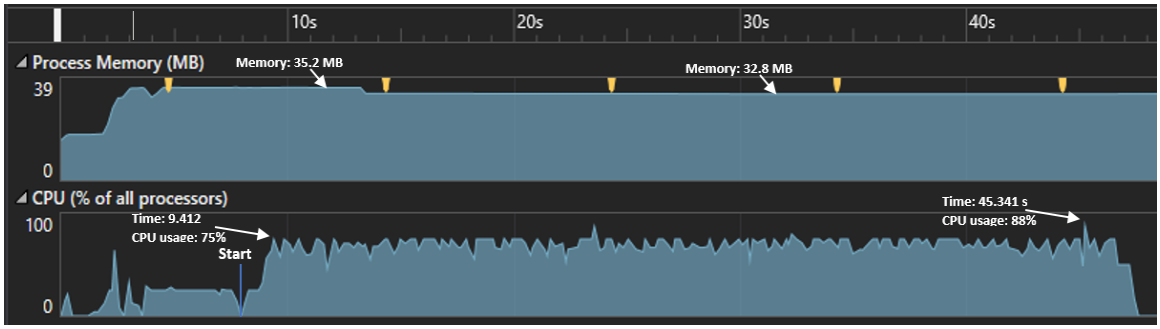

Figure 5: CPU and memory usage of SLA-LFSR

VII. CONCLUSION

The pseudorandom number generator is essential for cryptography, if it is compromised the entire cryptosystem will collapse. The design of PRNG should be a trade-off between speed and security to fulfill the need of security applications. The initialization or reseeding time should be minimized, and the periodic length should be long. The result clearly states that the initialization time, reseeding time and the need of frequent reseeding are the factors that drastically degrade the performance of PRNG. The

analysis shows that the SLA-LFSR is the optimal PRNG among all the compared generators due to its long periodic length, optimal memory and CPU usage along with improved security.

REFERENCES

[1] L. Blum, M. Blum and M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator", SIAM Journal on Computing, vol. 15, no. 2, pp. 364-383, 1986.

[2] Thomas W. Cusick, "Properties of the x2 mod N Pseudorandom Number Generator", IEEE Transactions on Information Theory, vol. 41, no. 4, July 1995.

[3] A. Sidorenko and B. Schoenmakers, "Concrete Security of the BlumBlum-Shub Pseudorandom Generator", Cryptography and Coding, LNCS, Springer vol. 3796, pp 355-375, Nov 2005.

[4] F. Montoya Vitini, J. Mufioz Masque and A. Peinado Dominguez, "Bound for linear complexity of BBS sequences", IEEE Electronics Letters, vol. 34, No. 5, April 1998. Temporal PRNG Aqeel Khalique IJCA

[5] Lehmer, D. H. 1949. Mathematical methods in largescale computing units. 2nd Symposium on Large-Scale Digital Calculating Machinery. pp. 141-146.

[6] S. Park and K. Miller, "Random number generators: good ones are hard to find", Communications of the ACM, vol. 31, no. 10, pp. 1192-1201, 1988.

[7] A. Klein, "Linear Feedback Shift Registers", Stream Ciphers, pp. 17-58, 2013.

[8] M. Alkhatib, "Acoustic Lightweight Pseudo Random Number Generator based on Cryptographically Secure LFSR", M.tech, Jamia Hamdard, 2017.

[9] M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800–22rev1a (2010).