International Journal of Computer Applications (0975 – 8887)
Volume 78 – No.16, September 2013

True Random Number Generator using Fish Tank Image

Rajat Katyal
University of Pune
S.C.O.E. – Dept. of I.T.
Vadgaon (Bk), Pune-41, India

Ankit Mishra
University of Pune
S.K.N.C.O.E. – Dept. of I.T.
Vadgaon (Bk), Pune-41, India

Adarsh Baluni
University of Pune
S.K.N.C.O.E. – Dept. of I.T.
Vadgaon (Bk), Pune-41, India

ABSTRACT

A Pseudo Random Number Generator (PRNG) uses a deterministic system and an initial seed to generate random numbers. In order for the output sequence to be truly random, a truly random input seed is used. Most True Random Number Generators (TRNG), use noise in the form nuclear decay, atmospheric noise, electrical noise or Brownian motion as their initial seed.

In order to reduce the computational complexity, we use a simple setup of a fish tank as the variable environment, capturing its images over time. The image data is then applied to a reduction algorithm and hash function to generate the initial seed. We propose a cost efficient method of extracting the true seed from the image data and applying it to a pseudo random generator, a Linear Congruential Generator (LCG) in our case to give true random numbers.

General Terms

PRNG, TRNG, LCG

Keywords

True random number generator, Image data

1. INTRODUCTION

Random numbers are the numbers that are part of a sequence in which values are uniformly distributed over a definite set and any future value cannot be reliably predicted based on the existing set of values. The sequence is produced by what is usually called as a random number generator. There are two types of random number generators: pseudo - random number generator and true random number generator. A pseudo - random number generator uses a deterministic algorithm which produces random numbers based on some initial value known as seed. Most random number sources actually utilize a pseudo-random generator. Since the output is purely a function of the seed data, the actual entropy of the output can never exceed the entropy of the seed [2].In certain real world secure systems the quality of random numbers is of paramount importance. For such applications it is not recommendable to use PRNGs as they use a deterministic algorithm. The output for such PRNGs are same for a particular seed, hence it is obvious that they cannot be secure if the seed is predictable in any which way.

TRNGs however use physical phenomena that are unpredictable in nature to generate true random numbers. The physical phenomena that are currently used are radioactive decay, atmospheric noise, nuclear decay, Brownian motion, clock drifts etc [9]. These results are then applied to cryptographic hash function for obtaining uniformly distributed outputs. Unfortunately one does not find these methods as peripherals for the modern day PC's making them very difficult to implement.

We have used a simple fish-tank setup as the variable environment and capturing its images over time as the initial seed. A different image is used every time to generate a set of random numbers. The image data provides the initial seed which is extracted using a reduce algorithm and hashing. This truly random seed is then fed to a pseudo random algorithm, a linear congruential generator in our case, to give a set of random numbers.

2. NEED

Random numbers are required in a lot of areas. Games, statistical sampling, Monte Carlo method simulation, gambling and lottery, sort and hash algorithms are a few of the fields that utilize the element of randomness.

The extensive utility of random numbers lies in the real - world security systems for exchange of classified data. Cryptography uses random numbers to a great extent for example while generating encryption key pairs and digital signatures requiring a high degree of randomness [10].

3. GENERATION

3.1 Existing Methodologies

The Hardware random number generators use different sources of entropy to calculate the initial seed values. These generally require complex hardware setups. Some of the commonly used methods are:

- Radioactive material decay time
- Atmospheric Noise
- Electrical noise from resistors or semiconductors

A similar approach to ours is to use a randomness extractor such as cryptographic hash function against certain non-uniformly random source.

This technique had been used by the Lavarnd generator that took images of the patterns of floating point materials in a lava lamp, extracting the seed and feeding it to a pseudo random number generator.

3.2 Our Approach

We have used a fish tank setup having over 20 live fishes as our variable environment source. Clicking its images over time we get the random and unpredictable image data. The image data is then applied to a reduce algorithm and hashing to generate a true and unpredictable seed.

This seed is fed to a pseudo random algorithm similar to that of a linear congruential algorithm in order to generate truly random numbers [14].

We have used the GIF image format having 640x480 pixels, in order to reduce the computational time and complexity. Two image samples are shown below along with their true seed values.

3.3 Samples

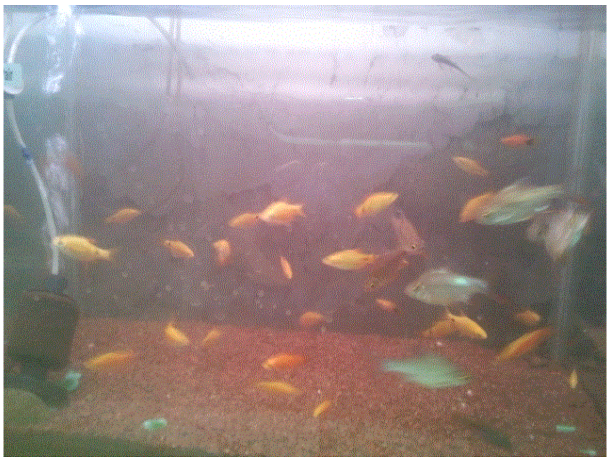
Fig 1: Sample Image A

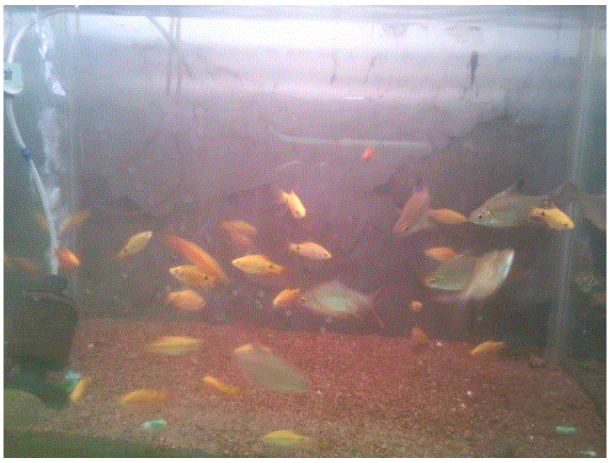
Fig 2: Image Sample B

In order to generate the different set of random numbers, we use a different image samples. For the shown image samples, the generated true initial seeds were:

Image A: 16338

Image B: 3453

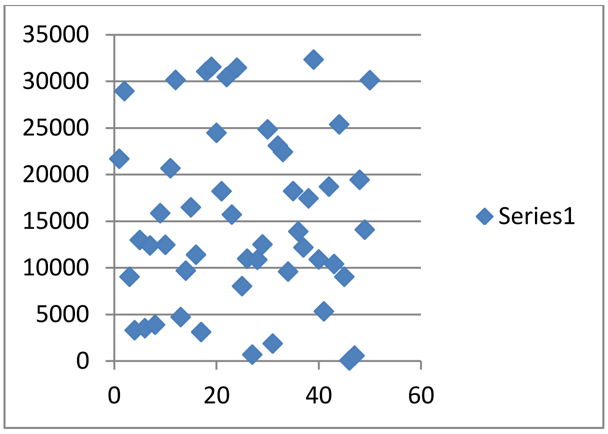

Fig 3: Sample set of 50 possitive random numbers

A small sample set of 50 random numbers produced using our generator is shown in Fig 3. The sample has a range of 0 - 32768. The figure illustrates the randomness of the generated numbers.

3.4 FLOWCHART

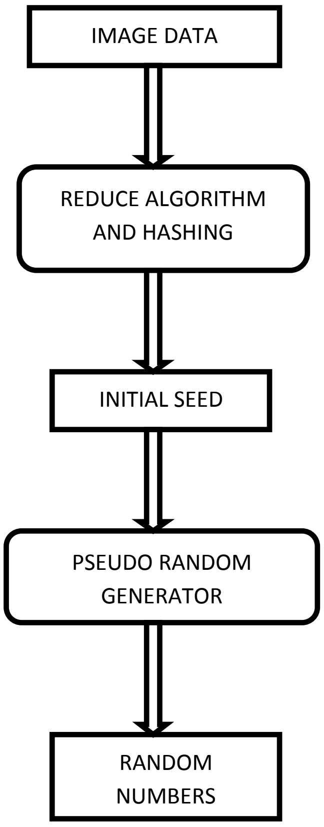

Fig 4: Flowchart of True Random Number Generator

4. TESTING

There are a wide variety of testing methods available that are carried out to check the quality of generated random numbers. These are typically applied to a PRNG for its quality check or any new type of random number generator [11].

Since we utilize an already proven pseudo random algorithm, a linear congruential generator, we did a quality test for our chosen variables and the initial seed. The observed pattern showed a high quality of randomness.

A single type of test cannot determine the quality of randomness. Also in a few tests, a vague pattern of result can actually mean higher quality of randomness as it is basically a factor of unpredictability.

A Runs Test carried out on 80 numbers (below Table 1) yields the following result. Such a test alone cannot be a quality testing factor for randomness, but is one of the essential tests.

Some other known testing methods are are frequency test, serial test, poker test and gap test.

Table 1: Runs Test carried on a sample of 80 numbers

7053	7716	1270	6327	3041
5846	3780	3705	444	6035
4682	6603	5612	8623	893
4319	8631	4384	980	4195
1618	5393	3641	7440	7430
7954	2538	5232	4324	5356
1919	320	8676	1102	2864
4830	5704	749	1773	3347
7545	8503	914	3770	1524
2219	4633	3774	7635	1939
63	2391	5476	1662	2766
2541	3267	7942	1667	8222
5249	1920	8396	1151	5693
1683	6473	3452	270	6502
1246	4585	290	2133	8577
7928	2106	69	7192	3981

Sample Set	**80**
Number of Runs(R)	**50**
P- Value	**0.02107**
Conclusion	**Moderate evidence against Sequential Randomness**

5. RESULT AND CONCLUSION

The random number generator is truly random as the seed value used is taken from an unpredictable source. Hence it completely avoids the possibility of determinism which is a risk factor in pseudo random number generators.

The generator yields truly random numbers up to 16 bits or 65536 with a high degree of randomness. The random number generation can be divided mainly into two phases: Image to seed conversion and Pseudo random generation.

It is also much easier to implement as compared to other hardware random number generators which use radioactive decay, atmospheric noise, electrical noise, etc. and require a much complex hardware setup to generate randomness. The TRNG using fish tank image data is a cost efficient alternative to generate truly random numbers.

6. LIMITATIONS AND FUTURE SCOPE

1. Image processing time period

Although the pseudo random processing time is comparable to any good PRNG but it involves an additional overhead of image processing time which makes it slightly slower.

The Future Scope of the True Random Number Generator is to reduce the overall processing time. This can be done by using a camera to generate the hexadecimal image data and using it directly, thereby reduce the GIF image conversion overhead.

2. Output range: 16bit

The 16bit range (0-65536) of random numbers can be increased by using a bigger sized compiler and micro-processor, while maintaining the high degree of randomness.

7. ACKNOWLEDGMENTS

Our thanks to P.Hellekalek and everybody that has contributed in generating good quality random numbers. Also special thanks to our friend Tejaswi Bhutada.

8. REFERENCES

[1] P. Hellekalek, Good random number generators are (not so) easy to find, Mathematics and Computers in Simulation 46 (1998) 485-505.

[2] Benjamin Jun and Paul Kocher, P. 1999 The Intel Random Number Generator. Technical Report. University of Maryland at College Park.

[3] N. K Pareek, V. Patidar, K. K Sud, A Rndom Bit Generator Using Chaotic Maps, International Journal of Network Security, Vol. 10, no. 1, pp. 32-38, 2010.

[4] Tang, H.C., “An Analysis of Linear Congruential Random Number Generators when Multiplier Restrictions Exist,” European Journal of Operational Research, Vol. 182, pp. 820828 (2007).

[5] Hedayatpour, S., Chuprat, Suriayati, Hash functions-based random number generator with image data source, Open Systems (ICOS), 2011 IEEE.

[6] Xuan Li, Guoji Zhang, Yuliang Liao, Chaos-based true random number generator using image

[7] P. Murali and R.Palraj, True Random Number Generator Based on Image for Key Exchange program.

[8] Xuan Li; Guoji Zhang; Yuliang Liao, "Chaos-based true random number generator using image," Computer Science and Service System (CSSS), 2011 International Conference on , vol., no., pp.2145,2147, 27-29 June 2011 doi: 10.1109/CSSS.2011.5974933

[9] Matsumoto, M. and T. Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator." ACM Transactions on Modeling and Computer Simulation 8, no. 1 (1998): 3|30.

[10] Volos, C.K.; Kyprianidis, I. M.; Stouboulos, I. N., "Image encryption process based on a chaotic True Random Bit Generator," Digital Signal Processing, 2009 16th International Conference on , vol., no., pp.1,4, 5-7 July 2009 doi: 10.1109/ICDSP.2009.5201107

[11] Gentle, J. E. Random Number Generation and Monte Carlo Methods, (2nd Ed.) SpringerVerlag, 2003.

[12] Geman, S. and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian

[13] Restoration of Images." IEEE Transactions on Pattern Analysis and Machine Intelligence 6, no. 6 (1984): 721|741

[14] Junod, P. "Cryptographic Secure Pseudo-Random Bits Generation: The Blum|Blum|Shub Generator." August 1999.http://crypto.junod.info/bbs.pdf