# An Efficient Pseudo Chaotic Number Generator Based on Coupling and Multiplexing Techniques

Ons Jallouli*, Safwan El Assad*, Mohammed Abu Taha *, Maryline Chetto†, René Lozi‡, Daniel Caragata§
*Institut d'Electronique et de Télécommunications de Rennes, UMR CNRS 6164, Université de Nantes, France,
Emails: {ons.jallouli, safwan.elassad, mohammad.abu-taha}@univ-nantes.fr
†Institut de Recherche en Communications et cybernétique de Nantes, UMR 6597, Université de Nantes, France,
Email: maryline.chetto@univ-nantes.fr
‡Laboratoire J.A. Dieudonné, UMR CNRS 7351, Université de Nice Sophia-Antipolis, France
Email:Rene.LOZI@unice.fr
§ Universidad Tecnica Federico Santa Maria, Department of Electronic Engineering, Av. Espana 1680, Valparaso, Chile
Email:daniel.caragata@usm.cl

*Abstract*—**Cryptosystems require a very good and secure source of randomness for their security. Such a source is extremely difficult to obtain in practice. Hence, developing a secure pseudo-random number generator reveals necessary. In this paper, we propose a new pseudo chaotic number generator. The proposed structure integrates three discrete chaotic maps: Piece Wise Linear Chaotic (PWLCM), Skewtent and Logistic maps, which are weakly coupled and implemented with finite precision N=32 bits. It also includes a chaotic multiplexing technique. The experiment results and statistical analysis prove the robustness of the proposed generator as well as its efficiency in terms of computation time in comparison with know chaotic generators of the literature. Based on the previous structure, a random number generator that uses Linux generator has been designed : `"/dev/urandom"`.**

***Keywords*–*Pseudo-chaotic number generator; Weakly Coupling; Chaotic multiplexing technique; Random numbers; Security analyses.***

## I. Introduction

Random numbers generators are useful for a variety of purposes in various contexts including statistical mechanics, gaming industry, cryptography and communications, etc. There are two basic types of random number generator: True Random Number Generators (TRNGs) and Pseudo-Random Number Generators (PRNGs). TRNGs produce a random bit stream from a non-deterministic natural source. They extract randomness from certain physical phenomena such as thermal and atmospheric noises. TRNGs are characterized by a higher security. However, their implementation requires additional devices, which make them inconvenient (cost and slow) [1]. A PRNG is a deterministic algorithm that produce numbers whose distribution is uniform, by inputting an initial seed (often generated by a TRNG). PRNGs are important in practice for their rapidity in number generation, reproducibility of the pseudo-random sequences and requiring less memory for storage [2].

Over the past years, many researchers have been attracted to chaos in the design of PRNGs, due to its intrinsic properties such as ergodicity, randomness and high sensitivity to initial conditions and parameters [3]. Several PRNGs have been proposed in the literature. Shujun et al. [4] presented a novel pseudo-chaotic bit generator based on a Couple of Chaotic Systems called CCS-PRBG. Analyses show that it has good cryptographic properties, but the speed of the proposed generator is not high enough for real time applications. Lozi [5] introduced new models for very weakly coupled logistic and tent maps using single or double precision numbers. Also, he used a double threshold chaotic sampling and mixing in weakly coupled tent maps [6]. This technique improves randomness of the generated sequence but causes a decrease in the speed performance. In [7], we proposed a Pseudo Chaotic Number Generator (PCNG) based on three weakly-coupled discrete skewtent maps and uses a chaotic multiplexing technique. This structure is very secure but its implantation was not optimized.

In this paper, we present an efficient PCNG based on three weakly coupled discrete chaotic maps namely PWLCM (Piece-Wise Linear Chaotic Map), Skewtent and Logistic maps. Besides, the structure uses a chaotic switching technique that increase the security performance. In addition, based on the proposed PCNG and Linux generator "/dev/urandom". The paper is organized as follows: the architecture of the proposed PCNG is described in Section II. In Section III, we give its performance in terms of security and computation time. Then, we present in Section IV the proposed RNG. Finally, we conclude our work in Section V.

## II. Structure of the proposed PCNG

The scheme of the proposed PCNG is presented in Fig. 1. It is based on iterating three chaotic maps, namely, Pwlcm, skewtent and logistic maps, which are weakly coupled by a coupling matrix. It also includes a chaotic multiplexing technique [6], [8]–[13] .

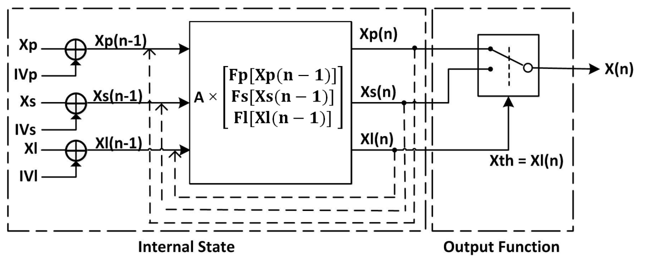

Figure 1: Structure of the proposed chaotic generator.

The generated samples X(n) are quantified on N = 32 bits.

The secret key of the system consists of:

- The initial conditions Xp, Xs and Xl of the three chaotic maps: Pwlcm, Skewtent and Logistic respectively, ranging from 1 to $2^N$-1.
- The control parameter Pp and Ps of Pwlcm and Skewtent maps, where $1 \leq Pp \leq 2^{N-1}$ and $1 \leq Ps \leq 2^N - 1$.
- The parameters of the coupling matrix A, $\varepsilon_{ij}$, ranging from 1 to $2^k$ with k $\leq$ 5.

The chaotic system uses three initial vectors IVp, IVs and IVl, each quantified on 32 bits.

All the initial conditions, parameters and initial vectors are chosen randomly from file "`/dev/urandom`", interfaces to the entropy pool sources on the Lunix kernel.

The functionality of the chaotic generator is as follows:

*Step1:* It calculates the initial values Xp(0), Xs(0) and Xl(0) as follows:

$$\begin{cases} Xp(0) = Xp \oplus IVp \\ Xs(0) = Xs \oplus IVs \\ Xl(0) = Xl \oplus IVl \end{cases}$$

*Step2:* The chaotic maps are weakly coupled by a coupling matrix A, as indicated by (1):

$$\begin{bmatrix} Xp(n) \\ Xs(n) \\ Xl(n) \end{bmatrix} = A \times \begin{bmatrix} Fp[Xp(n-1)] \\ Fs[Xs(n-1)] \\ Fl[Xl(n-1)] \end{bmatrix}. \tag{1}$$

where $A$ is given by:

$$A = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & \varepsilon_{22} & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & \varepsilon_{33} \end{bmatrix} \tag{2}$$

with

$$\begin{cases} \varepsilon_{11} = 2^N - \varepsilon_{12} - \varepsilon_{13}. \\ \varepsilon_{22} = 2^N - \varepsilon_{21} - \varepsilon_{23}. \\ \varepsilon_{33} = 2^N - \varepsilon_{31} - \varepsilon_{32}. \end{cases}$$

Fp[X(n-1)], Fs[X(n-1)] and Fl[X(n-1)] are the discrete PWLCM, Skewtent and Logistic maps functions respectively [7].

$$Fp[X(n-1)] =$$

$$\begin{cases} \left\lceil 2^N \times \frac{X[n-1]}{P} \right\rceil & \text{if } 0 < X[n-1] \leq P \\ \left\lceil 2^N \times \frac{X[n-1]-P}{2^{N-1}-P} \right\rceil & \text{if } P < X[n-1] \leq 2^{N-1} \\ \left\lceil 2^N \times \frac{2^N-P-X[n-1]}{2^{N-1}-P} \right\rceil & \text{if } 2^{N-1} < X[n-1] \leq 2^N - P \\ \left\lceil 2^N \times \frac{2^N-X[n-1]}{P} \right\rceil & \text{if } 2^N - P < X[n-1] \leq 2^N - 1 \\ 2^N - 1 - P & otherwise \end{cases} \tag{3}$$

$$Fs[X(n-1)] =$$

$$\begin{cases} \left\lfloor \frac{2^N \times X[n-1]}{P} \right\rfloor & \text{if } 0 < X[n-1] < P \\ 2^N - 1 & \text{if } X[n-1] = P \\ \left\lfloor \frac{2^N \times (2^N - X[n-1])}{2^N - P} \right\rfloor & \text{if } P < X[n-1] < 2^N \end{cases} \tag{4}$$

$$Fl[X[n-1]] =$$

$$\begin{cases} \left\lfloor \frac{X[n-1] \times [2^N - X[n-1]]}{2^N - 1} \right\rfloor & \text{if } X[n-1] \neq [3 \times 2^{N-2}; 2^N] \\ 2^N - 1 & \text{if } X[n-1] = [3 \times 2^{N-2}; 2^N] \end{cases} \tag{5}$$

*Step3:* The output samples X(n) are controlled by a threshold T and a chaotic sample Xth which is equal to Xl(n):

$$X(n) = \begin{cases} Xp(n), & \text{if } 0 < Xth < T \\ Xs(n), & \text{otherwise} \end{cases} \tag{6}$$

Notice that at the end of each execution, a new IV is generated. This allows to produce a new key stream for the next execution (using the same secret key).

## III. Simulation results and analysis

### A. Security Analysis

*1) Key space analysis:* A PCNG should have a large key space in order to make brute-force attacks infeasible. It is generally accepted that a key space of size equal or greater to $2^{128}$ is secure. The size of the secret key of the proposed system is:

$$|K| = (|Xp| + |Xs| + |Xl|) + (|Pp| + |Ps|) + 6 \times |\varepsilon_{ij}|. \tag{7}$$

where $|Xp| = |Xs| = |Xl| = |Ps| = 32$ bits ; $|Pp| = 31$ bits and $|\varepsilon_{ij}|$ is equal to 5 bits. Therefore $|K| = 189$ bits.

The size of the secret key is large enough to resist any brute-force attacks. Such a large space of keys is a necessary condition, but not sufficient. Indeed, the generated sequences must be cryptographically secure.

*2) Key Sensitivity:* The sensitivity on the key is an essential property for any PCNG. Naturally, a small change in the secret key causes a large change in the output sequences. In order to verify this characteristic, we calculate the Hamming Distance of two sequences generated with only one bit change (lsb bit) in the parameter Pp. We calculate the average Hamming Distance $D_{Hamming}$ between two sequences $S_1$ and $S_2$, over 100 random secret keys. The $D_{Hamming}(S_1, S_2)$ is defined by the following equation:

$$D_{Hamming}(S_1, S_2) = \frac{1}{Nb} \times \sum_{K=1}^{Nb} (S_1[K] \oplus S_2[K]) \tag{8}$$

where $Nb$ is the number of bits in a sequence. The obtained average value of Hamming distance is equal to 0.499988. This value is close to the optimal value of 50%. This result illustrates the heigh sensitivity on the secret key of the proposed PCNG.

## B. *Statistical Analysis*

To test the statistical properties of the proposed PCNG, we used several known statistical tests. They concern mapping, auto and cross-correlation, histogram, chi square and NIST test.

*1) Phase space trajectory or mapping analysis:* The mapping or the phase space trajectory of the generated sequences reflects the dynamic behaviour of the system. In Fig. 2, we give a zoom of the obtained mapping. It is messy due to the used chaotic coupling and switching techniques. Therefore, it is impossible to identify the type of the used chaotic maps.

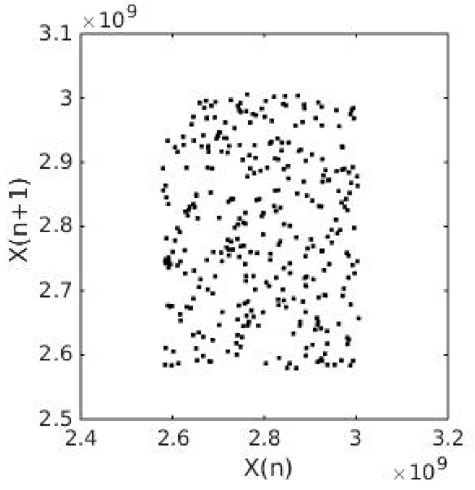

Figure 2: Mapping of the sequence $X$.

*2) Histogram and Chi-square analysis:* We study the distribution uniformity of the generated sequences. A PCNG must provide a uniform distribution in the whole phase space. We give in Fig. 3 the histogram of a generated sequence, by our PCNG, formed by $10^7$ samples.

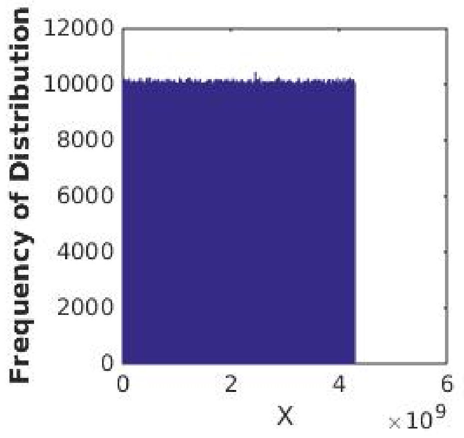

Figure 3: Histogram of a generated sequence $X$.

Visually, we observe that the generated sequence is nearly uniformly distributed. We then apply the chi-square test to assert the uniformity of the sequence [14]. The experimental Chi-square $\chi^2$ value is given by:

$$\chi^2_{\text{exp}} = \sum_{i=0}^{K-1} \frac{(O_i - E_i)^2}{E_i}. \qquad (9)$$

where $K$ is the number of classes (sub-intervals) chosen in our experiment equal to 1000, $O_i$ is the number of observed (calculated) samples in the i-th class and $E_i$ is the expected number of samples of a uniform distribution, $E_i = 10^7/K$.

We compare the experimental value given by (9) with a theoretical value obtained for a threshold $\alpha$=0.05 and a degree of freedom $K$-1=999. The experimental value of chi-square is equal to 1027.26. This value is lower than the theoretical one which is equal to 1073.64. These results confirm the uniformity of the generated sequence.

*3) Correlation analysis:* Correlation reflects the intensity of connection which may exist between two random variables. In PCNG, the values in the sequences must not be repeated nor correlated. To avoid the statistical analysis, the correlation coefficients of two sequences $X$ and $Y$, computed with nearby initial conditions, should be close to zero.

The correlation coefficient is calculated by the following equation:

$$\rho_{XY} = \frac{\sum_{i=1}^{N}(x_i - \bar{X})(y_i - \bar{Y})}{[\sum_{i=1}^{N}(X_i - \bar{X})^2]^{1/2} \times [\sum_{i=1}^{N}(Y_i - \bar{y})^2]^{1/2}}. \qquad (10)$$

where $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$ are the mean values of two sequences $X$ and $Y$ respectively.

The calculated correlation coefficient $\rho_{XY}$ is equal to 0.0022 (close to zero). Also, in Fig. 4 we give a zoom of the cross-correlation function of sequences $X$ and $Y$, and the auto-correlation of sequence $X$. Results clearly show the negligible correlation between the generated sequences $X$ and $Y$.

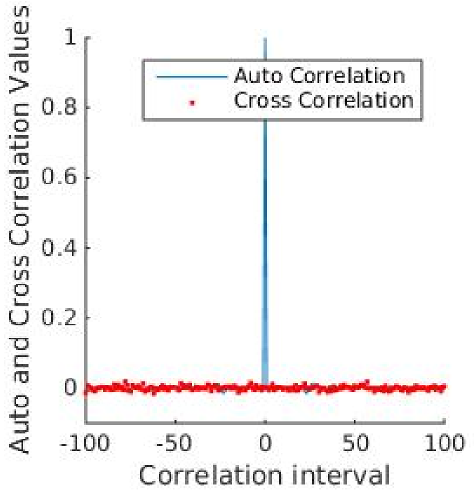

Figure 4: Cross-correlation of sequences $X$ and $Y$, and auto-correlation of sequence $X$.

*4) NIST test:* We apply the NIST statistical test, which presents one of the most popular standard test for analysing randomness of binary data. The STS 2.1.2 version statistical test suite published in [15] is used. It consists of a battery of

188 tests (globally 15 different tests) to conclude regarding the randomness or non-randomness of binary sequences. For each test, a set of m P-values are expected to indicate failure. Indeed, an $\alpha$ = 0.01, indicates that 1% of the m sequences are expected to fail.

- A $P - value \geq \alpha = 0.01$ would mean that the sequence would be random with a confidence of $(1 - \alpha) = 99\%$.
- A $P - value < \alpha = 0.01$ would mean that the conclusion was that the sequence is non-random with a confidence of $(1 - \alpha) = 99\%$.

In our experiments, we generate m = 100 sequences, each of length $10^6$ bits, and $\alpha = 0.01$.

The results are given in Fig. 5 and Table I. It can be seen that the bit-sequences pass all tests and fulfil the hypothesis of randomness. Therefore, the proposed chaotic generator is robust against statistical attacks.

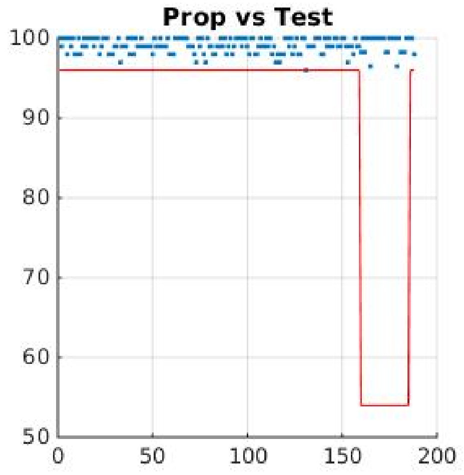

Figure 5: NIST tests results.

TABLE I: P-VALUES AND PROPORTION RESULTS OF NIST TEST.

| *Test* | *P-value* | *Proportion* |
|---|---|---|
| Frequency test | 0.972 | 99 |
| Block-frequency test | 0.055 | 99 |
| Cumulative-sums test | 0.834 | 98.5 |
| Runs test | 0.534 | 100 |
| Longest-run test | 0.290 | 99 |
| Rank test | 0.964 | 97 |
| FFT test | 0.384 | 97 |
| Non periodic-templates | 0.482 | 98.939 |
| Overlapping-templates | 0.637 | 98 |
| Universal | 0.237 | 98 |
| Approximate entropy | 0.936 | 99 |
| Random-excursions | 0.314 | 99.364 |
| Random-excursions-variant | 0.295 | 99.435 |
| Serial test | 0.606 | 100 |
| Linear-complexity | 0.290 | 99 |

## C. Speed Performance

Speed is an important factor for evaluating the performance of a PCNG. For the proposed PCNG, we calculate the bit rate (in Mega bits per second) and the number of needed cycles to generate one byte. All experiments are performed on a personal computer with Intel(R) Core(TM) i5-4300M CPU @2.60GHz and memory 15,6 GB and the operating system is Ubuntu 14.04 Trusty Linux distribution, using GNU GCC Compiler. In tableII, we give, over 100 different secret keys, the average Bit Rate in Mbps and the average number of needed cycles to generate one byte (NCpB) for the proposed PCNG to generate 31250 samples. And we compare the obtained results with some known generators. The Bit Rate and NCpB are calculated respectively as follows:

$$Bit\ Rate(Mbps) = \frac{Generated\ data\ size(Mbits)}{Average\ generation\ time(s)} \quad (11)$$

$$NCpB = \frac{CPU speed(Hz)}{Bit\ Rate(Byte/s)} \quad (12)$$

TABLE II: COMPUTING PERFORMANCE OF SOME KNOWN PCNGS.

| *Pseudo chaotic generator* | *Bit Rate (Mbps)* | *NCpB* |
|---|---|---|
| Proposed PCNG | 514.73 | 41 |
| Jallouli et al. [7] | 138 | 151 |
| Shujun et al. [4] | 9 | 711 |

We notice that the proposed chaotic generator is faster than the following known pseudo random number generator of the literature: Francois et al. [16], QUANTIS [17] and Blum Blum Shub [18].

## IV. Design of a random number generator

The ability to generate random numbers is important for many applications including cryptographic ones and others applications that do not require deterministic sequences when using the same secret key. For that, we adapt our PCNG, to be used as a random numbers generator. The proposed random numbers generator has the same structure of Fig. 1, but includes a refresh process repeated every R% samples, for generating a sequence X(n) of length n, to update the values of Xp(R), Xs(R) and Xl(R). The refresh process uses random values from the file ”/dev/urandom” [19], interface to the Linux kernel's random number generator. For example, when using R = 50%, the bit rate of the proposed random numbers generator is equal to 397 Mbits/s.

## V. Conclusion and future work

In this paper, we reported a work on the design, realization and test of a new pseudo chaotic number generator. This one is based on three discrete chaotic maps: PWLCM, Skewtent and Logistic that are weakly coupled. The proposed PCNG also includes a chaotic switching technique. Results of the statistical analysis show that the proposed PCNG has very good cryptographic properties due to its structure. In addition, it runs faster than other well known pseudo random number generators. Furthermore, the structure of the PCNG is updated to be used as a RNG for various applications that need random numbers such as generation of cryptographic keys, computer games and some classes of scientific experiments.

Our future work will focus on a software realization of chaos-based stream ciphers and the measurement of their energy consumption.

## VI. Acknowledgement

The authors would like to thank the European Program Erasmus Mundus E-GOV-TN.

## References

[1] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Transactions on computers, vol. 56, no. 1, 2007, pp. 109–119.

[2] H. Karimi, S. Morteza Hosseini, and M. Vafaei Jahan, "On the combination of self-organized systems to generate pseudo-random numbers," Information Sciences, vol. 221, 2013, pp. 371–388.

[3] L. Kocarev and S. Lian, Chaos-based cryptography: Theory, algorithms and applications. Springer, 2011, vol. 354.

[4] L. Shujun, M. Xuanqin, and C. Yuanlong, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography," in Progress in CryptologyINDOCRYPT 2001. Springer, 2001, pp. 316–329.

[5] R. Lozi, "Giga-periodic orbits for weakly coupled tent and logistic discretized maps," Modern Mathematical Models, Method and Algorithms for Real World Systems, A.H. Siddiqi, I.S. Duff and O.Christensen Editors, Anamaya Publishers, New Delhi India, 2006, pp. 80–124.

[6] ——, "Emergence of randomness from chaos," International Journal of Bifurcation and Chaos, vol. 22, no. 02, 2012, p. 1250021.

[7] O. Jallouli, S. El Assad, M. Chetto, R. Lozi, and D. Caragata, "A novel chaotic generator based on weakly-coupled discrete skewtent maps," in International Conference on Internet Technology and Secured Transactions, London, 2015, pp. 38–43.

[8] P. Amato, D. Mascolo, I. Pedaci, and D. Ruggiero, "Method of generating successions of pseudo-random bits or numbers," May 3 2006, uS Patent App. 11/381,474.

[9] R. Lozi, "New enhanced chaotic number generators," Indian Journal of Industrial and Applied Mathematics, vol. 1, no. 1, 2007, pp. 1–23.

[10] M. V. Petersen and H. M. B. Sørensen, "Method of generating pseudo-random numbers in an electronic device, and a method of encrypting and decrypting electronic data," Jan. 30 2007, uS Patent 7,170,997.

[11] S. El Assad, H. Noura, and I. Taralova, "Design and analyses of efficient chaotic generators for crypto-systems," in World Congress on Engineering and Computer Science 2008, WCECS'08. Advances in Electrical and Electronics Engineering-IAENG Special Edition of the. IEEE, 2008, pp. 3–12.

[12] S. El Assad, "Chaos based information hiding and security," in Internet Technology And Secured Transactions, 2012 International Conference for. IEEE, 2012, pp. 67–72.

[13] K. Desnos, S. El Assad, A. Arlicot, M. Pelcat, and D. Menard, "Efficient multicore implementation of an advanced generator of discrete chaotic sequences," in Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for. IEEE, 2014, pp. 31–36.

[14] L. Pace, "Chi-square tests," in Beginning R. Springer, 2012, pp. 217–228.

[15] B. Elaine and K. John, "Recommendation for random number generation using deterministic random bit generators," NIST SP 800-90 Rev A, Tech. Rep., 2012.

[16] M. François, T. Grosges, D. Barchiesi, and R. Erra, "A new image encryption scheme based on a chaotic function," Signal Processing: Image Communication, vol. 27, no. 3, 2012, pp. 249–259.

[17] A. K. Hartmann, Practical guide to computer simulations. World Scientific, 2009.

[18] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo random number generator," SIAM J. Comput., vol. 15, no. 2, May 1986, pp. 364–383. [Online]. Available: http://dx.doi.org/10.1137/0215025

[19] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the linux random number generator," in 2006 IEEE Symposium on Security and Privacy. IEEE, 2006, Berkeley/Oakland, CA, pp. 385 – 400.