A True Random Number Generator Based on Hyperchaos and Digital Sound

Je Sen Teh[1], WeiJian Teng[2] and Azman Samsudin[3]

[1,3] School of Computer Sciences
Universiti Sains Malaysia
Penang, Malaysia

[2] School of Computing and Engineering
Disted College
Penang, Malaysia

[1]jesen teh@hotmail.com, [2]weijian.teng@gmail.com, [3]azman.samsudin@usm.my

***Abstract*—True random number generators (TRNG) play an important role in many fields that require unpredictable and nondeterministic random number sequences. Unlike their pseudorandom counterparts, TRNGs are more computationally expensive as they need to harvest entropy from physical phenomena. To generate high quality true random numbers at a fast rate, this paper introduces a new TRNG based on hyperchaos and digital sound. The characteristics of a hyperchaotic map such as sensitivity to initial conditions and complex behavior amplifies noise obtained by sampling environmental sound through a computer microphone. The random numbers generated are then evaluated using statistical test suites such as NIST SP 800-22, DIEHARD and ENT. Because nondeterminism cannot be proved by merely running test suites, entropy analysis is performed to determine the unpredictability of these sequences. Results show that the proposed TRNG can generate true random numbers at a high rate while maintaining strong statistical quality. In addition, the entropy source requires only an inexpensive computer microphone which is already built into many laptops and handheld devices. Therefore, the proposed generator provides a low-cost and easily obtainable option for applications that require true random numbers.**

Keywords—True random number generator; audio sample; chaotic map; entropy; security verification;

I. INTRODUCTION

Random number generators (RNG) can be divided into two main categories depending on unpredictability: pseudorandom number generators (PRNG) and true random number generators (TRNG). A PRNG requires a seed value to generate random-like sequences mathematically. However, a PRNG will always produce the same sequence for the same seed. TRNGs instead extract entropy from unpredictable physical sources such as radioactive decay [1], electrical noise [2] or even thermal fluctuations [3] to produce random numbers that are nondeterministic.

RNGs are important in many fields such as cryptography, statistics, scientific research and even gaming. The selection between PRNG and TRNG is made depending on the requirement of a particular application. As an example, a TRNG is used to generate secret keys or nonces for cryptographic applications to ensure that they are highly unpredictable whereas PRNGs are preferred for situations where reproducibility is required. Compared to PRNGs, TRNGs are usually slower as they harvest entropy from physical phenomena. Recently, a number of software-based TRNGs have been introduced that attempt to extract randomness from computer behavior such as race conditions [4], air turbulence in disk drives [5] or even mouse movement [6]. These TRNGs have an advantage whereby existing computer hardware can be used to generate secure random sequences without additional cost.

In this paper, a new TRNG based on hyperchaos and digital sound is introduced. The source of randomness is noise from ambient sound that is recorded by a computer or device microphone. This noise is then amplified by using a hyperchaotic system with qualities such as random-like behavior, long period length and high sensitivity to slight changes to its inputs. These characteristics allows the system to magnify changes of even 1 bit to produce entirely different random number sequences. This entropy magnification is used to design a TRNG that is both fast and secure. There exists prior audio-based TRNGs in literature which can only achieve speeds of up to 21.6 kilobits per second (kbps) [7] and 4.41 kbps [8]. In addition, both proposals lack thorough security verification.

The rest of this paper is arranged as follows: Sect. II introduces basic concepts of chaos theory and audio sampling which will be used in the algorithm design described in Sect. III. Sect. IV and V discusses the security and performance of the proposed TRNG. Finally in Sect. VI, the paper is concluded with closing remarks.

II. PRELIMINARIES

A. Chaos Theory

A chaotic system is a mathematical function that depicts aperiodicity, sensitivity, ergodicity, diffusion and confusion characteristics that fulfill the requirements of cryptographic algorithms. Therefore, they have been used in the design of encryption algorithms [9], hash functions [10] and even key exchange protocols [11]. A popular chaotic map used in many designs is the chaotic tent map shown in (1) which maps the interval of [0,1] onto itself:

$$f_T(x) = \begin{cases} \alpha x & 0 \le x < 0.5 \\ \alpha(1-x), & 0.5 \le x \le 1, \end{cases} \quad (1)$$

where α is the control parameter and x is the current state of the chaotic system. The behavior of a chaotic map is determined by its control parameter value. In the case of the tent map, when α increases from 0 to 2, the chaotic map evolves from periodic to aperiodic. This can be seen from the

 978-1-5090-2549-7/16/$31.00 ©2016 IEEE

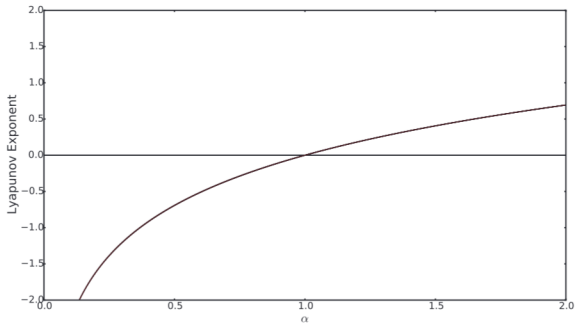

Fig. 1. Tent map Lyapunov exponent.

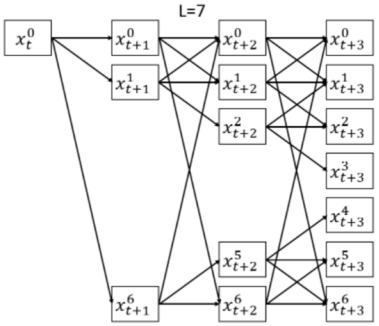

Fig. 2. CCML diffusion for $L = 7$

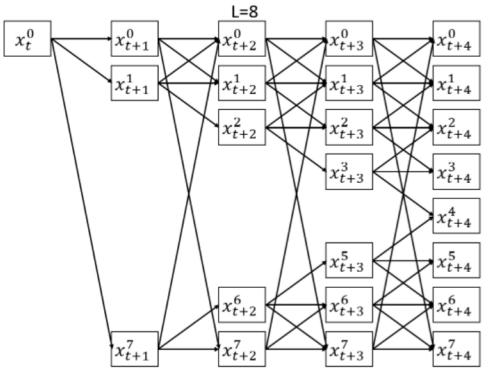

Fig. 3. CCML diffusion for $L = 8$

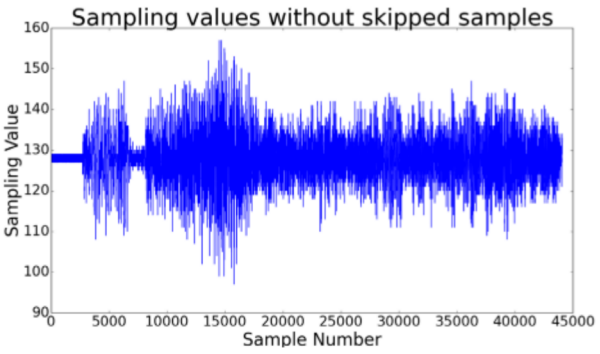

Fig. 4. Audio recording without transient elimination.

Lyapunov exponent plot of the tent map in Fig. 1 where a positive Lyapunov exponent indicates chaotic behavior.

The chaotic system used in the TRNG proposal is a spatiotemporal chaotic system known as a coupled chaotic map lattice (CCML). Compared to unimodal chaotic maps like the tent map, the CCML has longer period length, higher complexity and has multiple positive Lyapunov exponents [12]. A chaotic system with multiple positive Lyapunov exponents is also known as hyperchaotic. The CCML equation is as shown in (2):

$$x_{t+1}^{i} = (1-\epsilon)f(x_t^i) + \frac{\epsilon}{2}(f(x_t^{i+1}) + f(x_t^{i-1})), \qquad (2)$$

where ϵ is a coupling constant, $i = \{1, 2, 3, , L\}$ is the map index, L is the size of the system and $f(x)$ is a local chaotic map. In the proposed design, the tent map is used as the local map with the control parameter, $\alpha = 1.99999$ and the CCML coupling constant, $\epsilon = 0.05$.

For a single chaotic state to affect all other states, a total of $flr(\frac{L}{2})$ iterations are required, where the $flr(y)$ function rounds y downwards and returns the largest integral value that is not greater than y. Examples for $L = 7$ and $L = 8$ are shown in Fig. 2 and 3 respectively.

B. Audio Sampling

The source of randomness for the proposed TRNG is ambient sound which is recorded by a microphone and digitized. Recordings are obtained using the Waveform Audio Interface API at a sampling rate of 44.1 kHz. Each sample is then stored as 8-bit variables. In order to eliminate the transient effect that occurs at the start of a recording, the first 10000 samples are discarded. The audio recording in Fig. 4 depicts the transient effect, where there exists a period of non-randomness prior to the 5000th sample. An audio recording that skips the first 10000 samples is shown in Fig. 5 where the transient effect has been eliminated.

Based entropy assessment, three least significant bits (LSBs) depict sufficient uncertainty to be used in the TRNG. Three types of environments were tested: quiet (muffled microphone), normal and noisy. In these environments, the entropy for a 1-Mbit sequence produced by concatenating three LSBs from each sample is calculated. The results were 3.871733, 6.703212 and 7.165172 bits per byte for quiet, normal and noisy environments respectively. This entropy will then be amplified using hyperchaos.

III. Algorithm Description

As discussed in the previous section, extracting three LSBs from each audio sample can only achieve the highest entropy of 7.165172 bits per byte. To achieve maximal entropy, the randomness of the audio clip is amplified using hyperchaos. The CCML's states, x_t^i are represented by the 64-bit IEEE double-precision binary floating-point (FP) format. With the CCML size as $L = 8$, the size of the entire CCML state is $L \times 64 = 512$ bits.

Before performing iterations of (2), bits from the audio samples are used to perturb the x_t^i states. Perturbation of a chaotic state refers to a modification of the chaotic map's initial

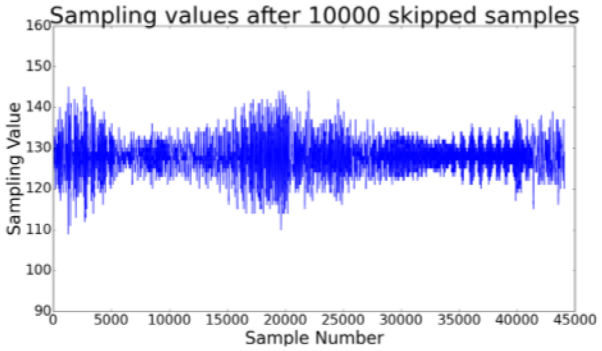

Fig. 5. Audio recording with transient elimination.

state by an external source. In the case of the proposed TRNG, the external source are bits from the audio samples. Three bits from each sample is used to perturb one chaotic state, therefore a total of eight samples ($8 \times 3 = 24$ bits) are used at one time. The required length of the true random number sequence (TRNS) is used to determine how many audio samples are recorded.

The interval between each perturbation operation is selected based on the number of rounds required to diffuse the bits in one state to all eight states. For $L = 8$, the number of rounds required is $flr(\frac{8}{2}) = 4$, therefore perturbation of the x_t^i only occurs once every four iterations. The value of x_t^i is modified based on (3):

$$x_t^i = ((0.071428571 \times r^y) + x_t^i) \times 0.666666667, \quad (3)$$

where r^y is the value of the 3-bit random number obtained from the y-th audio sample. This equation is chosen such that the value of x_t^i will be modified by r^y but still remain within the interval of [0,1] as required for the iteration of the chaotic tent map.

The overall flow of the generator's algorithm is as follows: After the length of the required TRNS is determined, an audio clip is recorded using the microphone and post-processed to extract three bits from each sample as identified in Sect. III. The states for the CCML, $x_0^i, i = \{0, , 7\}$ are initialized with the arbitrary FP values (0.141592, 0.653589, 0.793238, 0.462643, 0.383279, 0.502884, 0.197169, 0.399375). These values are perturbed based on (3) before the CCML is iterated four times based on (2). This ensures that the perturbation of each chaotic state has been diffused to all eight states as depicted in Fig. 3. Let $z^i, i = \{0, ..., 7\}$ be the 64-bit binary representations of the final eight FP states, $x_4^i, i = \{0, ..., 7\}$. The values of z^i are used to produce four 64-bit numbers based on (4):

$$\begin{aligned} z^0 &= z^0 \boxplus swap(z^4) \\ z^1 &= z^1 \boxplus swap(z^5) \\ z^2 &= z^2 \boxplus swap(z^6) \\ z^3 &= z^3 \boxplus swap(z^7) \end{aligned} \quad (4)$$

where $\boxplus$ is a bitwise exclusive-or (XOR) operation and $swap()$ function swaps the 32 most significant bits (MSBs) with the 32 least significant bits. This is performed because the first 12 MSBs represent the sign bit and exponent bits which have low randomness. By performing the swap followed by an XOR operation, the TRNG makes full use of the entire 64-bit values to produce outputs. The XOR operation also contributes to reducing bias in the final output, O which is a 256-bit random number sequence obtained by bitwise concatenation of z^i values ($O = z^0|(|z^1|)|z^2||z^3$). The entire TRNG is summarized in Algo. 1.

Algorithm 1 TRNG Algorithm

Input: Size of required TRNS (bits): N, CCML state variable: **double** x_t^i, CCML size: L, number of iterations: γ, coupling constant: ϵ, audio sample 3-bit numbers: r^y, output file size (bits): α.
Output: Array of 256-bit random numbers: O

$n = \frac{N}{256} \times 8$
$A = rec(n)$ //Record audio clip with n-samples and store in array A
for $i = 0$ **to** n **do**
  $r^i = extract(A[i])$ //Extract 3 bits from each sample
end for
$x_0^0 = 0.141592, x_0^1 = 0.653589, x_0^2 = 0.793238, x_0^3 = 0.462643$
$x_0^4 = 0.383279, x_0^5 = 0.502884, x_0^6 = 0.197169, x_0^7 = 0.399375$
$c = 0$ //Set a counter to 0
while $sizeof(O) \leq N$ **do**
  for $i = 0$ **to** $(L - 1)$ **do**
    $x_t^i = ((0.071428571 \times r^y) + x_t^i) \times 0.666666667$
    $c = c + 1$
  end for
  for $t = 0$ **to** $(\gamma - 1)$ **do**
    for $i = 0$ **to** $(L - 1)$ **do**
      $x_{t+1}^i = (1 - \epsilon) f_T(x_t^i) + \frac{\epsilon}{2}(f_T(x_t^{i+1 \bmod L}) + f_T(x_t^{i-1 \bmod L}))$
    end for
  end for
  for $i = 0$ **to** $(L - 1)$ **do**
    $z^i = uint64(x_4^i)$ //Cast x_4^i as an unsigned 64-bit integer z^i
    $x_0^i = x_4^i$ //Set initial value for next round of iterations
  end for
  for $i = 0$ **to** $(\frac{L}{2} - 1)$ **do**
    $z^i = z^i \boxplus swap(z^{i+\frac{L}{2}})$
  end for
  $O \leftarrow z^0||z^1||z^2||z^3$ //Concatenate and store 256-bit random number
end while

IV. Security Analysis

The TRNG design is analyzed based on the evaluation criteria for physical RNGs proposed by Schindler and Killmann [13]. The evaluation criteria is based on standardization efforts by the German Department for Security in Information Security Technology BSI, whereby two functionality classes of RNGs are introduced: P1 and P2 generators. The requirement for P1 generators is to be statistically inconspicuous (passes statistical test suites) whereas P2 generators are required to have forward and backward security in addition to fulfilling the P1 requirement. In this section, the proposed TRNG is shown to fulfill both P1 and P2 requirements.

A. Statistical Test Suites

The proposed TRNG is evaluated using three test suites, the NIST SP 800-22 [14], DIEHARD [15] and ENT [16]. Although test suites cannot differentiate between true random and pseudorandom, they are vital to detect any statistical defects that exist in a generator's outputs.

The NIST test suite consists of 15 statistical tests that each produce a P-value as outputs. These tests were performed on 1000 different 1-Mbit samples. The minimum passing ratio, p' for each test is 0.980561 for all the tests except the random excursion test which requires $p' = 0.978060$. p' is calculated based on a significance level of 0.01 for 1000 samples. In addition, an overall P-value is calculated based on the 1000

TABLE I. NIST TEST RESULTS

Test Name	P-value	Passing Ratio	Result
Frequency	0.796268	0.991	Pass
Block Frequency	0.437274	0.990	Pass
Cumulative Sums	0.395940	0.993	Pass
Runs	0.678686	0.990	Pass
Longest Run	0.064822	0.987	Pass
Rank	0.467322	0.992	Pass
FFT	0.643366	0.987	Pass
Non-overlapping Templates	0.987896	0.982	Pass
Overlapping Templates	0.117432	0.984	Pass
Universal	0.919131	0.987	Pass
Approximate Entropy	0.461612	0.994	Pass
Random Excursions	0.967842	0.980	Pass
Random Excursions Variant	0.002472	0.988	Pass
Serial	0.322135	0.984	Pass
Linear Complexity	0.154629	0.993	Pass

TABLE II. DIEHARD TEST RESULTS

Test Name	P-value	Result
Birthday Spacings Test	0.064600	Pass
Overlapping 5-Permutation Test	0.125674	Pass
Binary Rank Test for 31x31 Matrices	0.877028	Pass
Binary Rank Test for 32x32 Matrices	0.656309	Pass
Binary Rank Test for 6x8 Matrices	0.829382	Pass
Bitstream Test	0.01281	Pass
Overlapping-Pairs-Sparse-Occupancy Test	0.0680	Pass
Overlapping-Quadruples-Sparse-Occupancy Test	0.1047	Pass
DNA Spacings Test	0.0448	Pass
Count-The-1s Test on a Stream of Bytes	0.936210	Pass
Count-The-1s Test for Specific Bytes	0.031879	Pass
Parking Lot Test	0.100920	Pass
Minimum Distance Test	0.902581	Pass
3D-Spheres Test	0.418768	Pass
Squeeze Test	0.922773	Pass
Overlapping Sums Test	0.477592	Pass
Runs Test	0.880336	Pass
Craps Test	0.906958	Pass

individual P-values of each test, where a minimum of P=0.0001 indicates a uniform distribution of the 1000 P-values. The NIST results in Table I indicates that the proposed TRNG successfully passes the NIST test suite.

The DIEHARD test suite has 18 statistical tests that produce P-values ranging from [0,1]. A 87.5-Mbit file is generated as the input sample for the DIEHARD test. This file size is based on the samples provided by the test suite. If a P-value is below 0.01 or higher than 0.99, the generator has failed a test. The DIEHARD results in Table II only lists the worst case results for tests with multiple P-values. The proposed TRNG successfully passes all tests.

The third test suite used in the evaluation of the TRNG is the ENT test suite which consists of 5 statistical tests with different measures for randomness. The results of testing is shown in Table III, where the experimental results is compared against the ideal results for each test. The generator is able to achieve near-ideal results, thus passing all tests successfully. In particular, the output of the generator is able to achieve near-maximum entropy with 7.999998 bits per byte.

TABLE III. ENT TEST RESULTS

Test Name	Test output	Ideal value	Result
Entropy	7.999998	8	Pass
Chi-square	44.19 %	50%	Pass
Arithmetic Mean	127.5184	127.5	Pass
Monte Carlo value for Pi	3.140933713	3.141592653	Pass
Serial correlation coefficient	0.000376	0	Pass

B. Analysis of Nondeterminism

The previous subsection has shown that the proposed TRNG has sufficient statistical randomness to fulfill the requirement for P1 generators. However, passing statistical test suites does not prove that a generator is nondeterministic as even good PRNGs can pass test suites. A nondeterministic generator is defined to be one that is able produce unpredictable outputs on each run, even when all the initial conditions or seed is the same.

To depict nondeterminism in the proposed generator,the following experiment is performed as proposed in [4]: a 1-Megabyte (MB) sequence is first generated using a *generate-reset* procedure. The *generate-reset* procedure requires that a generator is reset to its initial state after generating one block of random numbers. In the case of the proposed TRNG, each block is 256 bits. The *generate-reset* procedure is repeated 32768 times to produce a 1-MByte sequence ($\frac{1024\times1024\times8}{256} = 32768$). The Average Shannon Entropy (ASE) is then calculated for this random sequence based on (5):

$$H(X) = \lim_{k\to\infty} -\frac{1}{k} \sum_{\sigma\in\{0,1\}^k} P(\sigma_k) log_2 P(\sigma_k), \qquad (5)$$

where $P(\sigma_k)$ is the probability mass function of σ_k and the summation consists of the entire random number sequence, where all binary k-tuples of the form $\sigma_k = b_0, , b_{k-1}, b \in 0, 1$ are collected. In the proposed generator, each number is represented by one byte, therefore $k = 8$. The maximal ASE is equal to one. With the same initial conditions, a PRNG will generate the same random number repeatedly if the generate-reset procedure is performed, therefore failing this test. The ASE of the 1-MByte file produced by the proposed TRNG is $H(X) = 0.9836$. This result suggests that the TRNG is nondeterministic in nature.

Next, the TRNG is analyzed for backward and forward security to achieve the P2 property for generators. This characteristic is paramount to ensure that an attacker cannot predict previous or future outputs of a generator given the ability to corrupt the internal state of a generator [17]. To depict backward and forward security, evidence of an increase in entropy per internal random number that is sufficiently large [13] is required.

For a fixed block size, the Coron test [18] can be used to calculate the exact information of entropy per bit. However, the generator must first show to be at least Markovian (independent and memoryless) before the Coron test can be applied. The proposed TRNG already depicts the Markovian property from the result of the generate-reset test, so the Coron test can be performed directly. The suggested parameters for the test are $L = 8$, $Q = 2560$ and $K = 256000$. To fulfill the security criteria, the result of the test must be $f > 7.976$. The proposed

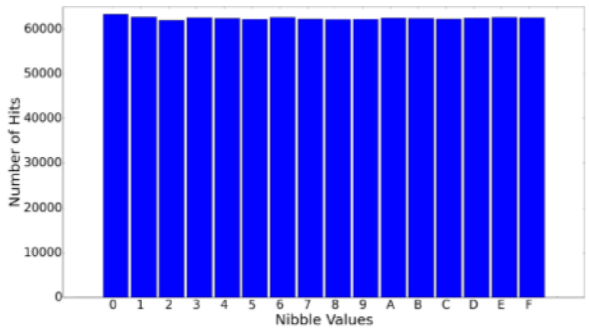

Fig. 6. Distribution of nibbles.

TRNG obtains a result of $f = 7.9977$. Coupled with the result of the generate-reset test, this suggests that the proposed TRNG has forward and backward security. Therefore, the generator successfully fulfills the criteria for P2 generators and can be used for applications that require high security.

C. Sensitivity and Distribution

In the proposed TRNG, the hyperchaotic CCML acts as a post-processing mechanism that amplifies any unpredictability obtained from audio samples as well as eliminate any bias in the final random sequence to obtain an evenly distributed output. To analyze the sensitivity of the CCML to slight changes, two 1-Mbit sequences are generated by the proposed TRNG, one using a simulated audio signal of all zero bits $(0, 0, 0, , 0)$ and the other with a simulated audio signal that starts with 1 followed by all zero bits $(1, 0, 0, , 0)$. The exclusive-or (XOR) difference between these two samples is taken as the error which is then measured by calculating its entropy. With even a slight input difference of one bit, the resulting error entropy of the proposed TRNG is 7.99889 bits per byte which is near maximum.

Next, a histogram plot is used to analyze the distribution of the TRNG. First, a 4000000-bit sequence is generated and divided into 4-bit values (nibbles). A total of 1000000 nibbles are plotted as shown in Fig. 6. It can be seen that all the nibble values are evenly distributed with equal probability of occurring. When analyzed at the bit level, the ratio between 0 and 1 bits is 1999661:2000339 where a 0 and 1 occurs with a probability of 49.99% and 50.01% respectively. Based on both these observations, it can be concluded that the generator is evenly distributed and has no detectable bias.

V. Performance Analysis

The performance of the proposed TRNG is heavily dependent on the size of the random number sequence that needs to be generated. If the time taken to record the audio samples is taken into account, the proposed TRNG can achieve a speed of 1.293 Mbit per second (Mbps). Compared to other audio-based TRNGs, the proposed generator's speed is approximately 61 times faster than [7] and 299 times faster than [8].

In practice, it is also possible to pre-record audio samples and store them in an entropy pool which will eliminate the recording overhead. This increases the TRNG's speed to approximately 186.052 Mbps.

VI. Conclusion

In this paper, a new TRNG based on hyperchaos and digital sound was proposed. A hyperchaotic system called the CCML was used to amplify noise obtained from audio samples which were recorded by a computer microphone. The proposed TRNG was able to pass the NIST, DIEHARD and ENT test suites, therefore has no detectable statistical defects in its output. In addition, the generator was proven to be nondeterministic while having forward and backward security. Its output sequences have no significant bias as they are evenly distributed even at the bit level. In terms of performance, the proposed TRNG outperformed other sound-based TRNGs and is able to achieve a throughput of approximately 1.293 Mbps. In summary, the proposed TRNG is a low-cost and easily accessible source of TRNs that is suitable for practical applications due to its security and speed.

Acknowledgment

This work has been supported by Fundamental Research Grant Scheme (FRGS - 203/PKOMP/6711427) funded by the Ministry of Higher Education of Malaysia (MOHE).

References

[1] J. Walker, *Hotbits: Genuine random numbers, generated by radioactive decay*, online at www.formilab.ch/hotbits, 2001.

[2] O. Creţ, T. Győrfi and A. Suciu, *Implementing true random number generators based on high fanout nets*, Romanian Journal of Information Science and Technology, vol. 15, no. 3, 2012, pp. 277-298.

[3] B. Jun and P. Kocher, *The Intel random number generator*, 1999.

[4] J.S. Teh, A. Samsudin, M. Al-Mazrooie, and A. Akhavan, *GPUs and chaos: a new true random number generator*, Nonlinear Dynamics, vol. 82, no. 4, July 2015, pp. 1913-1922.

[5] D. Davis, R. Ihaka and P. Fenstermacher, *Cryptographic randomness from air turbulence in disk drives*, Proc. Advances in Cryptology (CRYPTO 94), Lecture Notes in Computer Science, vol. 839, July 2001, pp. 114-120.

[6] Q. Zhou, X. Liao, K. Wong, Y. Hu and D. Xiao, *True random number generator based on mouse movement and chaotic hash function*, Information Sciences, vol. 179, no. 19, September 2009, pp. 3442-3450.

[7] N.G. Bardis, A.P. Markovskyi, N. Doukas and N.V. Karadimas, *True random number generator based on environmental noise measurement for military applications*, Proc. 8th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA 09), pp. 68-73.

[8] R. Morrison, *Design of a true random number generator using audio input*, Journal of Craptology, vol. 1, no.1, June 2001.

[9] S. Behnia, A. Akhshani, H. Mahmodi and A. Akhavan, *A novel algorithm for image encryption based on mixture of chaotic maps*, Chaos, Solitons & Fractals, vol. 35, no. 2, January 2008, pp. 408-419.

[10] J.S. Teh, A. Samsudin, A. Akhavan, *Parallel chaotic hash function based on the shuffle-exchange network*, Nonlinear Dynamics, vol. 81, no. 3, August 2015, pp. 1067-1079.

[11] M.S. Farash and M.A. Attari, *Cryptanalysis and improvement of a chaotic map-based key agreement protocol using Chebyshev sequence membership testing*, Nonlinear Dynamics, vol. 76. no. 2, April 2014, pp. 1203-1213.

[12] Y. Wang, L. Luo, Q. Xie and H. Tian, *A fast stream cipher based on spatiotemporal chaos*, Proc. International Symposium on Information Engineering and Electronic Commerce (IEEC 09), pp. 418-422.

[13] W. Schindler and W. Killmann, *Evaluation criteria for true (physical) random number generators used in cryptographic applications*, Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES 02), Lecture Notes in Computer Science, August 2002, pp. 431-449.

[14] A. Rukhin, J. Soto and J. Nechvatal, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, NIST Special Publication 800-22, National Institute of Standards and Technology, 2010.

[15] G. Marsaglia, *DIEHARD: A battery of tests of randomness*, online at http://stat.fsu.edu/pub/diehard/, 1996.

[16] J. Walker, *ENT Program*, online at www.fourmilab.ch/random, 2008.

[17] Y. Dodis, D. Pointcheval, S. Ruhault, D. Vergniaud and D. Wichs, *Security analysis of pseudo-random number generators with input: /dev/random is not robust*, Proc. ACM SIGSAC Conference on Computer & Communications Security (CCS 13), pp. 647-658.

[18] J-S. Coron, *On the security of random sources*, Public Key Cryptography, Lecture Notes in Computer Science, vol. 1560, October 1999, pp. 29-42.