Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 931802, 9 pages
doi:10.1155/2012/931802

Research Article

A Novel True Random Number Generator Based on Mouse Movement and a One-Dimensional Chaotic Map

Wang Xingyuan, Qin Xue, and Teng Lin

Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China

Correspondence should be addressed to Wang Xingyuan, wangxy@dlut.edu.cn

Received 14 July 2011; Revised 19 October 2011; Accepted 26 October 2011

Academic Editor: Stefano Lenci

We propose a novel true random number generator using mouse movement and a one-dimensional chaotic map. We utilize the x-coordinate of the mouse movement to be the length of an iteration segment of our TRNs and the y-coordinate to be the initial value of this iteration segment. And, when it iterates, we perturb the parameter with the real value produced by the TRNG itself. And we find that the TRNG we proposed conquers several flaws of some former mouse-based TRNGs. At last we take experiments and test the randomness of our algorithm with the NIST statistical test suite; results illustrate that our TRNG is suitable to produce true random numbers (TRNs) on universal personal computers (PCs).

1. Introduction

Random number generators (RNGs) have been widely used in recently science and technology, such as simulation, sampling, numerical analysis, computer programming, decision making, recreation, cryptographic protocols, and cryptosystems [1–7]. RNGs have two basic types: true random number generators (TRNGs) and pseudorandom number generators (PRNGs). TRNGs produce true random numbers, which are nondeterministic. That means even if all the previous values have been gotten, the next value is unpredicted. PRNGs, on the contrary, are deterministic. The pseudorandom numbers (PRNs) are generated by deterministic programs and an input called seed and can be predicted by post-time series. Because TRNGs have better security property, in the higher security required areas, like generating cryptographic keys and initialization variables in cryptographic protocols, and so forth, TPRGs are irreplaceable.

However, it is widely known that TRNGs need a certain physical phenomena, like thermal noise [8, 9], atmospheric noise [10, 11], and coin tossing [1]. So if one wants to utilize the above methods to produce true random numbers (TRNs) in a personal computer (PC), which is the most widely used platform, additional equipments must be required. It is impossible in short-finance situation. So we turn to figure out the problem by using the existed equipments. We consider mouse a proper source for TRNs, because it is used universally and it produces a real random source. People move mouse in a true random pattern, and others could hardly tell the regulation of it. Even the attackers get the pattern of the past mouse movement, they cannot tell the following action the user will take. Thus, mouse appears to be a TRNG, and an increasing number of researches are focusing at this field.

Hu et al. [12] and Zhou et al. [13] talk about the algorithms of generating TRNs based on mouse movement and chaotic cryptography, respectively. In literature [12], Hu et al. provide three approaches to postprocess the mouse movement pattern, and one way to transform mouse movement pattern to digital numbers. Experiment results show that their methods pass statistical tests. However, in literature [13], Zhou et al. also provide three approaches, two of which are different from the approaches in literature [12], while one of which is the same. Although two of approaches in literature [12] and all of those in literature [13] can pass the randomness test, there are some drawbacks of their algorithms. Firstly, their methods require proper movement numbers of mouse movement pattern; neither too much or too little will break the security of the algorithm. This is because their postprocess way is generally a way to count pixels of the image, so if the image's pixels' distribution is not uniform, the produced TRNs are not uniform, either. Thus, the produced RNs cannot pass statistical tests [13] and be provided randomness. That is to say, the TRNs' properties rely on the mouse-produced image's properties. Combining this reason and the issue of key space, literature [12] requires the number of points of samples to be between 256 and 1024, which limits user's action, and is inconvenient. Second, the "MASK" method, which is considered by that first paper as the best approach among the three methods of that paper and concluded that it is able to practice in common PC applications, is a parallel image encryption algorithm [13]; it is not suitable for using on a one-element PC. Plus, the tent-map-based approach (TMA) contained in literature [13] is evaluated by the author that is unconvinced for users, since it can only produce a 104-bit random number with a single mouse movement. As well as the other method mentioned in literature [13], the free forward-feedback nonlinear digital filter method (FFNF) is also not suitable for usage because of its low speed. And the discrete 2D chaotic map permutation method in literature [12] has not also been considered random at all. Thus, we just need to compare the left two approaches, the Spatiotemporal chaos method in literature [12] and the New approach based on tent map method (NPTM) in literature [13].

Here we propose a simple algorithm of TRNGs based on user's mouse movement and a one-dimensional chaotic map. We utilize the x-coordinate to be the length of an iteration segment of our TRNs and y-coordinate to be the initial value of this iteration segment. And, when it iterates, we perturb the parameter. We take some experiments and a NIST statistical suite to test the randomness of this TRNG and compare it to the two methods mentioned above. Results show that our algorithm is random and most of the randomness properties are better than the two methods. Plus, the time cost of our algorithm is even lower. And also, our algorithm needs no additional equipments. Therefore, we can conclude that our algorithm is an effective and practicable method to produce TRNs for universal computers.

2. Introduction of the One-Dimensional Chaotic Map

In 2009, Aguirregabiria proposed a class of one-dimensional smooth map [14], which is depicted as follows. It has a good property that, in a certain parameter interval, this class of maps has positive Lyapunov exponent:

$$f_r(x) = \frac{(1+r)f(x)}{f(c)+rf(x)}, \tag{2.1}$$

here f should accord in three conditions:

(i) $f : [0,1] \to [0,1]$ of class C^3,

(ii) the Schwarzian derivative is negative in the whole interval. Schwarzian derivative is defined as

$$Sf(x) = \frac{f'''(x)}{f'(x)} - \frac{3}{2}\left(\frac{f''(x)}{f'(x)}\right)^2, \tag{2.2}$$

(iii) f is a unimodal map and $f(0) = f(1) = 0$. That is to say, the map increases from $f(0) = 0$, when it comes to the maximum $f(c)$, $c \in (0,1)$, the map begins to decrease until $f(1) = 0$ again.

When map satisfies the three conditions above, we call the map "S-unimodal" [14].

Take $f(x) = x(1-x)$, for example, and in the rest of this paper, $f(x)$ is also defined as that. Then, by the proof in literature [14], the function $f_r(x)$ has positive Lyapunov exponent in the interval $(-0.75, +\infty)$. We draw the Lyapunov exponent in Figure 1(a) by computing numerically by

$$\lambda = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \ln\left|f_r'(x_n)\right|. \tag{2.3}$$

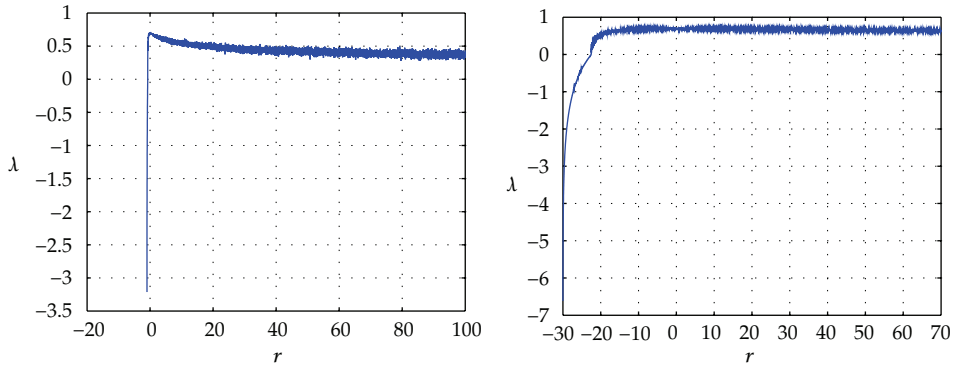
And then, we use a modified class of one-dimensional maps in (2.4), here k is the expansion coefficient of r on negative axle:

$$f_{rk} = \frac{(k+r)f(x)}{kf(c)+rf(x)}. \tag{2.4}$$

By simple calculation, one can find out that when $r > k$, f_{rk} is S-unimodal, too. Here we take $k = 30$; one can see that f_{rk} has positive Lyapunov exponent λ showed in Figure 1(b), when $r > -22.5$. And in the rest of this paper, we are likely to use these parameters to build our TRNG.

3. TRNGs Algorithm Based on Mouse Movement

Here we are going to depict our algorithm carefully. We firstly choose the one-dimensional map depicted before to be the iteration map of our TRNGs algorithm. We then get the pattern



(a) Map (1) for $r \in (-1, 100)$ (b) Map (4) for $r \in (-30, 70)$, $k = 30$

Figure 1: The curves of Lyapunov exponent.

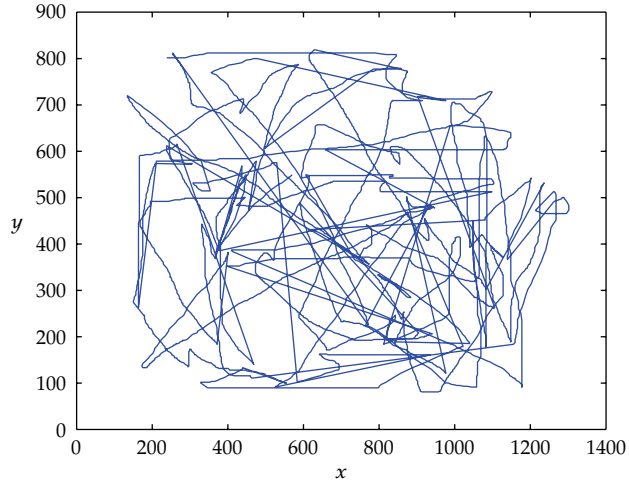


Figure 2: The mouse movement pattern.

of the mouse movement showed in Figure 2, which is the material of our TRNGs. We utilize the points of mouse movement, which construct Figure 2. We consider the x-coordinate of a point to be the numbers of the iteration and the y-coordinate to be the initial of the iteration. So that one point can deduce a sequence of numbers, which is produced by iterations. The more points the mouse moving, the more sequences will be produced. And as the initial value and the numbers of one iteration are totally unknown and absolutely cannot be predicted, those sequences are TRNs. And even there are only a few points existed; there will be lots of numbers produced. That solves the problem in literature [12] of constraining the number of points of mouse movement pattern. To remark it, the y-coordinate should be divided by 900 before it is used to be the initial value.

What is more, we add perturbation in each iteration, which will increase the randomness and break the periodic phenomenon. We replace the parameter r_i with the value $x_i - 1.5$. That will guarantee the parameter r will not depart too much but approximate the value -1. It will preserve properties of the TRNs we produced. And we use the binary

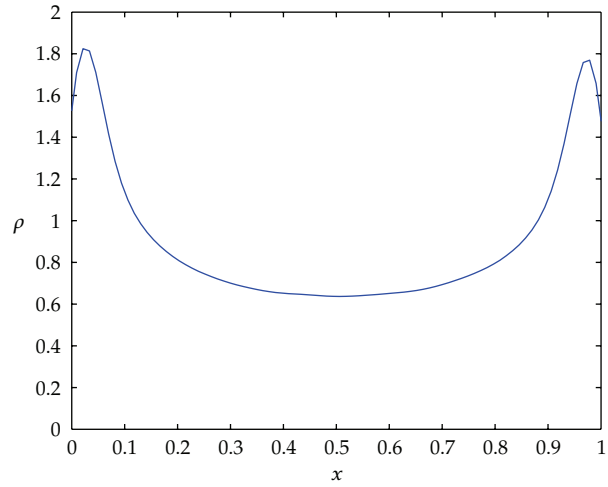


Figure 3: Kernel density map.

quantization to transform our TRNs to a binary sequence in (3.1). Here t is the threshold, and we adopt $t = 0.5$. Although there are lots of methods to be the transformation [15–18], we still use binary quantization for its simplicity. In the next section, we will introduce several experiments and statistical tests to test the sequence we generated whether random or not and compare the time cost and the randomness with the spatiotemporal chaos approach:

$$x_i' = \begin{cases} 0, & x_i < t, \\ 1, & x_i \geq t, \end{cases} \quad i \in N. \tag{3.1}$$

4. Experiments and Results

4.1. Kernel Density Map, Histogram, and Autocorrelation Function

Using the algorithm we proposed above and the pattern we have, we produced 1000000 TRNs. We firstly draw the kernel density map in Figure 3 and histogram in Figure 4 with the real value sequence of our TRNs, which are real numbers. The histogram equation is in

$$D = \sum_{0}^{n-d-1} x_i' \oplus x_{i+d}', \quad d \geq 0. \tag{4.1}$$

Considering x' is an n number binary sequence, x'' is a shifted sequence referred to x'. Let D be the number of bit-to-bit disagreements between $x'$and x''. Here d is the number of time sequences between x' and x''.

Moreover, we draw the autocorrelation function of our binary sequences TRNs using the equation in (4.2).

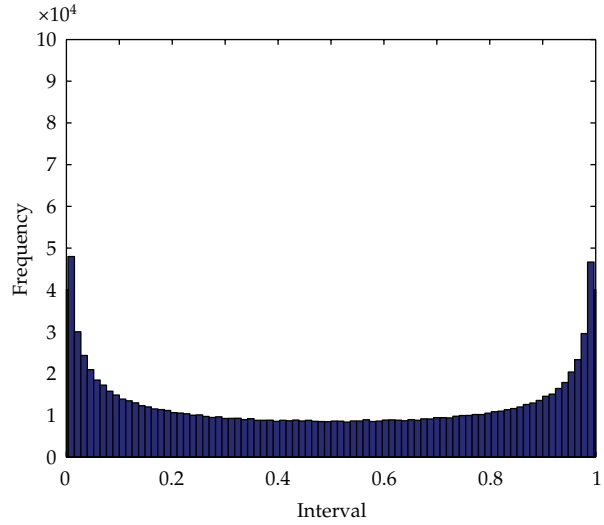


Figure 4: Histogram of the sequence.

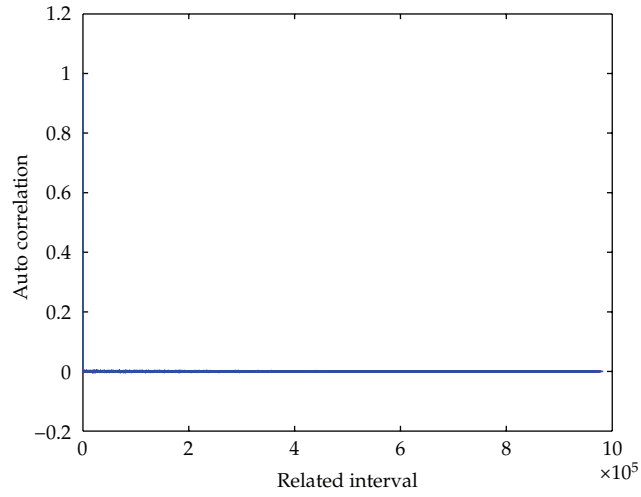


Figure 5: Autocorrelation function.

The autocorrelation function C is

$$C = \frac{n - d - 2D}{n - d}. \tag{4.2}$$

The autocorrelation function then is shown in Figure 5, which is a δ-like function.

4.2. NIST Statistical Test Suite

We also test our binary sequences with the NIST statistical test suite [19], which depicts deviations of a binary sequence from randomness. Unlike the Version 1.7 (and before)

Table 1: Average time required to generate a random number using different approaches.

Approach	Our algorithm	Spatiotemporal chaos	NPTM
Total time (milliseconds)	217	230	282

Table 2: 800-22 test's results.

Test	Our algorithm	Spatiotemporal chaos (n = 10000)	Spatiotemporal chaos (n = 1000)	NPTM	Conclusion
Frequency test	0.723674	0.3295	0.0938	0.1223	Random
Frequency test within a block	0.777709	0.8303	0.9175	0.2236	Random
Runs test	0.603211	0.9881	0.8999	0.0391	Random
Test for the longest run of ones in a block	0.784089	0.3782	0.2019	0.0616	Random
Binary matrix rank test	0.26483	0.8274	0.6724	0.3221	Random
Discrete fourier transform test	0.807748	0.2087	0.0346	0.2622	Random
Maurer's "universal statistical" test	0.441263	0.2457	0.1466	0.0640	Random
Linear complexity test	0.626767	0.5725	0.4654	0.3267	Random
Nonoverlapping template matching test	0.904832	0.6238	0.5383	0.0012	Random
Overlapping template matching test	0.625651	0.2264	0.0141	0.4118	Random
Approximate entropy test	0.70213	0.5039	0.4180	0.9737	Random
Serial test	0.617075	0.3448	0.1057	0.6434	Random
Cumulative sums test	0.737518	0.2788	0.1314	0.1875	Random
Random excursions test	0.72264	0.3538	0.4072	0.6352	Random
Random excursions variant test	0.863832	0.7459	0.6731	0.9216	Random

NIST statistical test suite [20] used in [12], which contains 16 items, including Lempel-Ziv complexity test (LZCT), we use the 2010 version NIST statistical test suite [19], the number of the items of which is 15. Thus, in this paper, we just list 15 items for comparison. The value of each test represents the degree of randomness of the tested sequence. If the value is bigger than 0.01, it demonstrates that the sequence passes the test and could be considered as random. And the bigger the value is, the more random the sequences are. For more details, please refer to literature [19]. In this test, all approaches were implemented with nonoptimized Matlab codes, running on an ordinary PC with 1.5 GHz Intel Celeron CPU.

In Table 1, we compare the time used in order to generate 256-bit data, including our algorithm, Spatiotemporal chaos in literature [12], and the New approach based on tent map (NPTM) in literature [13]. Moreover, we list results of the NIST statistical test suite of our algorithm, the spatiotemporal chaos approach contained in literature [12], and the New approach based on tent map (NPTM) contained in literature [13] in Table 2. As there are two tests of spatiotemporal chaos approaches for n = 1000 and n = 10000, respectively, we list them all in the table.

4.3. Results and Comparison

By comparing the items in Table 2, we find out that there are more items, although not all of them, of our algorithm that are better than that of the other three methods. For example, comparing to Spatiotemporal chaos (n = 10000) and Spatiotemporal chaos (n = 1000), there are 12 items in 15 in the table of our method that are better than that of the two methods. And also, comparing to NPTM, there are also 11 items of ours that are better than that of it.

5. Conclusion

In this paper, we first summarize some drawbacks of two proposed mouse movement TRNGs and propose a novel TRNG which conquers the flaws of the former two. The new algorithm is based on mouse movement and a one-dimensional chaotic map. The approach utilizes the x-coordinate of the mouse movement to be the length of the iteration, and the y-coordinate to be the initial value of this iteration segment. And we perturb the parameter with the real value of the produced TRNG itself when it iterates.

And then, we do some experiments and we compare the time cost of the three approaches and get the result that ours is a little bit faster than the other two. Last but not least, we test the three sequences with the NIST statistical test suite. Results show that our algorithm is better than the other two and is suitable to produce TRNs on universal PC.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (nos. 61173183, 60973152, and 60573172), the Superior University Doctor Subject Special Scientific Research Foundation of China (no. 20070141014), and the Natural Science Foundation of Liaoning province (no. 20082165).

References

[1] D. E. Knuth, *The Art of Computer Programming. Vol. 2: Seminumerical Algorithms*, Addison-Wesley, Reading, Mass, USA, 2nd edition, 1981.
[2] G. Jakimoski and L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps," *IEEE Transactions on Circuits and Systems. I*, vol. 48, no. 2, pp. 163–169, 2001.
[3] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the Association for Computing Machinery*, vol. 21, no. 2, pp. 120–126, 1978.
[4] B. Wang, Q. Wu, and Y. Hu, "A knapsack-based probabilistic encryption scheme," *Information Sciences*, vol. 177, no. 19, pp. 3981–3994, 2007.
[5] F. Cao and Z. Cao, "A secure identity-based proxy multi-signature scheme," *Information Sciences*, vol. 179, no. 3, pp. 292–302, 2009.
[6] D. Xiao, X. Liao, and S. Deng, "A novel key agreement protocol based on chaotic maps," *Information Sciences*, vol. 177, no. 4, pp. 1136–1142, 2007.
[7] D. Xiao, X. Liao, and S. Deng, "Using time-stamp to improve the security of a chaotic maps-based key agreement protocol," *Information Sciences*, vol. 178, no. 6, pp. 1598–1602, 2008.
[8] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 78–85, 2008.
[9] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo, "A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 403–409, 2003.

[10] D. Davis, R. Ihaka, and P. Fenstermacher, "Cryptographic randomness from air turbulence in disk drives," *Advances in Cryptology*, vol. 839, pp. 114–120, 1994.
[11] W. T. Holman, J. Alvin Connelly, and A. B. Dowlatabadi, "An integrated analog/digital random noise source," *IEEE Transactions on Circuits and Systems I*, vol. 44, no. 6, pp. 521–528, 1997.
[12] Y. Hu, X. Liao, K. W. Wong, and Q. Zhou, "A true random number generator based on mouse movement and chaotic cryptography," *Chaos, Solitons and Fractals*, vol. 40, no. 5, pp. 2286–2293, 2009.
[13] Q. Zhou, X. Liao, K.-W. Wong, Y. Hu, and D. Xiao, "True random number generator based on mouse movement and chaotic hash function," *Information Sciences*, vol. 179, no. 19, pp. 3442–3450, 2009.
[14] J. M. Aguirregabiria, "Robust chaos with variable Lyapunov exponent in smooth one-dimensional maps," *Chaos, Solitons and Fractals*, vol. 42, no. 4, pp. 2531–2539, 2009.
[15] Z. Hong and L. Xieting, "Generating chaotic secure sequences with desired statistical properties and high security," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 7, no. 1, pp. 205–213, 1997.
[16] P. Li, Z. Li, S. Fettinger, Y. Mao, and W. A. Halang, "Application of chaos-based pseudo-random-bit generators in internet-based online payments," *Studies in Computational Intelligence*, vol. 37, pp. 667–685, 2007.
[17] C. Ling and S. G. Sun, "4-phase spreading sequences by chaotic maps for CDMA," *Journal of China Institute of Communications*, vol. 19, no. 3, pp. 40–44, 1998.
[18] T. Kohda and A. Tsuneda, "Statistics of chaotic binary sequences," *IEEE Transactions on Information Theory*, vol. 43, no. 1, pp. 104–112, 1997.
[19] NIST, "A statistical test suite for random and pseudo-random number generators for cryptographic applications," 2010, http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf.
[20] "NIST Special Publication 800-22," 2001, http://csrc.nist.gov/rng/rng2.html.