2022 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2022, Virtual, December 12-15, 2022

Implementation of Secure and Fast Pseudo-Random-Number Generator on GPU

Hitoaki YOSHIDA†, and Takeshi MURAKAMI†

†Iwate University
18-33 Ueda, Morioka, Iwate 020-8550, Japan
Email: hitoaki@iwate-u.ac.jp, mtakeshi@iwate-u.ac.jp

Abstract– A novel iteration method with the modified activation function has been proposed to implement the secure and fast pseudo-random number generator (PRNG). The modified activation function has accelerated the pseudo-random-number (PRN) generation rate to 56.8 Tb/s by using NVIDIA GPU (A100). The PRNG consists of 1.67×10^7 Chaotic and Random Neural Networks (CRNNs), and the whole period of the PRN series is estimated at $P \approx 10^{100000000}$ based on the L.C.M. of the 10000 PRN series. The result is expected to apply the information security of the Controller Area Network (CAN) and that of the Autonomous GPU Accelerated systems, especially intelligent vehicle systems.

1. Introduction

Recently, the information security of the Controller Area Network (CAN) has attracted considerable attention [1].

We have reported the secure and fast cipher system by using the PRNG based on fixed-point arithmetic (Q5.26) [2- 5] and the application to the secure CAN [6].

NVIDIA has developed the Autonomous GPU Accelerated system (AGX), which is designed and built for all types of autonomous systems, including robotaxis. The NVIDIA DRIVE Orin SoC is expected as the central computer for intelligent vehicles, which is the integrated GPU that is equipped with Ampere architecture CUDA cores and TensorCores. The security system on GPUs is rapidly increasing in importance.

This study aims to implement a faster PRNG by using an optimized algorithm for the GPU architecture, moreover, to implement a securer PRNG by extending the period of the PRN series, extremely. The PRN series are expected as one-time random numbers for disposable use.

2. Results and Discussion

2.1. Basics of CRNN

The network composed of 4 artificial neurons (N_0-N_3) in the discrete-time system has been used for the PRNG (Figure 1 and Equation 1) [2-5]. I_j is an external input of the jth neuron (j = 0,1,2,3). An output of the jth neuron at time t+1 is defined as equation 1. w_{ij} is a synaptic weight between the ith neuron and jth neuron (i = 0,1,2,3), x_i (t) is an output from the ith neuron at time t and f is the asymmetric piecewise-linear-function (APLF3) (Figure 2). 7-bit-rotate-left instruction is additionally executed before the iteration only for x_1 and x_2.

The time series generated from CRNN can separate into 2 independent subseries: the α subseries and the β subseries [2,7]. In other words, 2 independent subseries are simultaneously generated (Table 1).

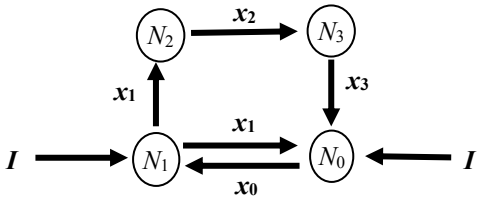

Figure 1: Chaotic and Random Neural Network (CRNN).

$$x_j(t+1) = f\left(\sum_{i=1}^{n} w_{ij}x_i(t) + I\right) \qquad (1)$$

Computer-generated chaotic time series is *eventually* periodic by the calculation with finite precision within our knowledge. A perturbation I_D (a randomly decided small floating-point value) is added to an external input I at an odd discrete time. I_D leads the 2 subseries to different periodic trajectories [2,7]. Outputs of each subseries (α and β) are shown in Table 1, where an output $x_i(t)$ is replaced with $\alpha_i(t)$ or $\beta_i(t)$ as to the corresponding subseries, and β-subseries are shown in a gray cell (Table 1).

Table 1: Outputs of Neuron as to Each Subseries. [a)]

Time	N_1	N_2	N_3	N_0
t-1	$\beta_1(t\text{-}1)$	$\alpha_2(t\text{-}1)$	$\beta_3(t\text{-}1)$	$\alpha_0(t\text{-}1)$
t	$\alpha_1(t)$	$\beta_2(t)$	$\alpha_3(t)$	$\beta_0(t)$
t+1	$\beta_1(t+1)$	$\alpha_2(t+1)$	$\beta_3(t+1)$	$\alpha_0(t+1)$
t+2	$\alpha_1(t+2)$	$\beta_2(t+2)$	$\alpha_3(t+2)$	$\beta_0(t+2)$
t+3	$\beta_1(t+3)$	$\alpha_2(t+3)$	$\beta_3(t+3)$	$\alpha_0(t+3)$

a) The output $x_i(t)$ in Equation 1 is replaced with $\alpha_i(t)$ or $\beta_i(t)$ as to the corresponding subseries. 2 subseries (α and β) don't mix with each other.

ORCID iDs First Author: 0000-0002-2255-6937,
Second Author: 0000-0002-3189-3265

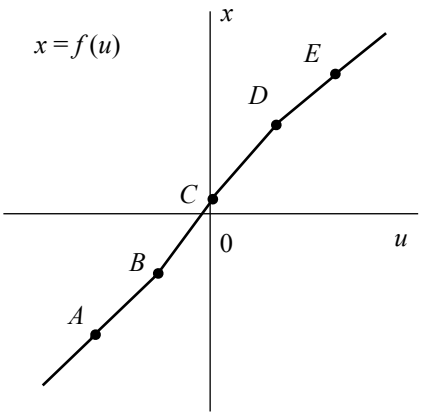

Figure 2: Activation Function *f* (APLF3).
A (-31.0001, -31.001), *B* (-7.9811, -8.29999), *C* (0.0001, 0.500012), *D* (7.981101, 8.6901), *E* (31.0002, 31.00999).

2.2. New Type Activation Functions for CRNN

A bottleneck process of the CRNN program in GPU is APLF3 because it involves conditional branches. While the slope of APLF3 is almost 1, that is, *df*/*du* = 0.9862 in (*A*, *B*), 1.1026 in (*B*, *C*), 1.0262 in (*C*, *D*), 0.9696 in (*D*, *E*), it may be regarded as random number addition. APLF3 may be replaced by the function in Equation 2, where *pr* is a PRN.

$$f(u) = u + pr \tag{2}$$

Conditional sentences (*e.g.*, if statements) in the program of CRNN may be replaced by the addition of a PRN, which is extracted from an output of CRNN by using the extraction method shown in Figure 3 [10]. Let us represent the extraction method of PRN by function $g(\,)$. Equations 1 and 2 are rewritten as Equations 3-6 using subseries, which shows only the iteration for α-subseries, and those for β-subseries are essentially the same except for a time lag by a unit time. Let us call the iteration described as Equations 3-6 *Method-3*.

$$\alpha_1(t) = w_{01}\alpha_0(t-1) + g(\alpha_2\,(t-1)) + I \tag{3}$$

$$\alpha_2(t+1) = w_{12}\alpha_1(t) + g(\alpha_1(t)) \tag{4}$$

$$\alpha_3(t) = w_{23}\alpha_2(t-1) + g(\alpha_0(t-1)) \tag{5}$$

$$\alpha_0(t+1) = w_{30}\alpha_3(t) + \mathrm{w}_{10}\alpha_1(\mathrm{t}) + g(\alpha_3(t)) + I \tag{6}$$

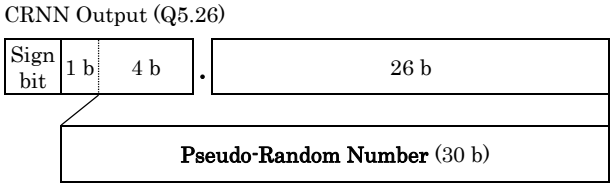

Figure 3: PRN Extraction Method from CRNN Output.

Finally, PRN series are extracted from the subseries by using the method shown in Figure 3, again.

In the preceding work, the values of another subseries were used as the additional PRN (Equations 7-10, *Method-2* in ref. [10]), the periods of the obtained PRN series, however, were too long to compute exact values. Therefore, the security of the PRN series could not be evaluated quantitatively based on the periods as shown in Figure 4. It is a fatal disadvantage for security applications.

$$\alpha_1(t) = w_{01}\alpha_0(t-1) + g(\beta_3\,(t-1)) + I \tag{7}$$

$$\alpha_2(t+1) = w_{12}\alpha_1(t) + g(\beta_0(t)) \tag{8}$$

$$\alpha_3(t) = w_{23}\alpha_2(t-1) + g(\beta_1(t-1)) \tag{9}$$

$$\alpha_0(t+1) = w_{30}\alpha_3(t) + \mathrm{w}_{10}\alpha_1(\mathrm{t}) + g(\beta_2(t)) + I \tag{10}$$

To reduce the period of the obtained PRN series, the additional pseudo-random number *pr* is generated from the same α-subseries in *Method-3* instead of the β-subseries.

An absolute value of *pr* may be important because a larger *pr* increases the probability of overflow. As for the piecewise functions of APLF3, the intercepts of the vertical axis are more than -0.43 and less than 0.96, however, the smaller *pr* (experimentally more than 4-bit-right-shifted *pr*) gave an unacceptable result, that is, the fail rate of "Frequency Test within a Block" [8-9] raised remarkably. The most suitable selection of *pr* in Equation 2 will be a problem in the future. The method shown in Figure 3 has been better so far.

2.3. Property of PRN Series from CRNN by using New Type Activation Function

The property of obtained PRN series by using the proposed method in this work is confirmed with "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications" (NIST SP800-22 test suite) [8-9]. The representative results are shown in Table 2 for "Proportion of Sequences Passing a Test" and in Table 3 for "Uniform Distribution of P-values Test".

Table 2: Result of "Proportion of Sequences Passing a Test" on the NIST SP800-22 Test Suite.

reference	**FR**	**FB**	**CS**	**RU**	**LR**	**RK**	**OT**
Method-3	0.2	0.1	0.1	0.3	0.3	0.0	0.8
Method-2 [10]	0.1	0.2	0.2	0.3	0.1	0.1	0.8

Table 3: Result of "Uniform Distribution of P-values Test" on NIST SP800-22 Test Suite.

reference	**FR**	**FB**	**CS**	**RU**	**LR**	**RK**	**OT**
Method-3	0.1	0.0	0.0	0.0	0.0	0.1	0.0
Method-2 [10]	0.0	0.0	0.1	0.0	0.1	0.0	0.1

The tests are repeated by 1000 times and obtained fail rates (%) are listed in the Tables. Abbreviations of test names in the tables are as follows: FR; Frequency Test, FB; Frequency Test within a Block, CS; Cumulative Sums Test, RU; Runs Test, LR; Test for the Longest Run of Ones in a Block, RK; Binary Matrix Rank Test, OT; Overlapping Template Matching Test.

2.4. Rate of PRNG by GPU (NVIDIA A100)

The modified CRNN has been implemented with CUDA 11.6 on a PC mounted with a GPU (NVIDIA A100). The result of the PRN generation rate is shown in Table 4 with the result in ref. [2] for comparison.

The rate of PRNG with the previous APLF3 has been accelerated by the latest GPU and CUDA from 2.85 Tb/s to 4.76 Tb/s (Table 4). Moreover, using the new type of activation function in *Method-3,* the rate become more than 10 times faster and reached 10 Tb/s (10^{13} b/s) order of magnitude. The result with *Method-2* is also shown in Table 4.

The number of threads (the number of CRNNs) is optimized to realize the maximum rate. As for the results of *Method-3* in Table 4, the number of threads is 1.67×10^7 (the exact value is 16777216). Therefore, $2\times1.67\times10^7$ subseries of the modified CRNN with different parameters work in parallel on A100. The whole period of CRNNs (P) can be defined as a period of a high-dimensional vector that consists of PRN series as elements [2]. The whole period of CRNNs is estimated in the next section.

Table 4: Maximum Rate of PRNG.

Activation Function [a]	Number of Threads [b]	Rate [c] / Tb s^{-1}	GPU [d]	CUDA Version
APLF1	6.61×10^7	1.78 [e]	P100	8.0
APLF3	6.61×10^7	2.85 [e]	P100	8.0
APLF3	1.63×10^4	4.76	A100	11.6
Method-3	1.67×10^7	56.78	A100	11.6
Method-2	1.67×10^7	57.62	A100	11.6

a) Types of activation functions.
b) The number of threads at the maximum rate of PRN generation.
c) The maximum rate of PRN generation in Tb s^{-1} (= 10^{12} b s^{-1}).
d) The product name of the NVIDIA GPU.
e) Data in ref. [2].

2.5. Estimation of Whole Period based on the Periods of PRN Series of CRNNs

The computer-generated chaotic time series is *eventually* periodic by the computation with finite precision within our knowledge. The computer-generated chaotic time series also stays in a transient trajectory before entering a periodic trajectory [4,5,7]. Time series staying a transient trajectory are useful as well as time series staying a periodic trajectory. In this study, however, the time in a transient trajectory is omitted simply for computing a period of time series.

The computed periods of the 10000 PRN series from CRNNs were obtained within the range 7.0×10^5 - 2.3×10^{10}. The 10000 PRN series consists of 5000 α-subseries and β-subseries. The whole period is estimated with L.C.M. (Least Common Multiple) of periods of each PRN series. As the transient trajectory is smaller than 4×10^{10}, after $t = 4\times10^{10}$ a period of the PRN series can be calculated without the influence of transient trajectories. The L.C.M. of the periods of the PRN series has been calculated precisely with a BigInteger class of Java language [2,12]. The L.C.M of the 10000 PRN series has been calculated step-by-step (Figure 4). Figure 4 shows the relationship between the calculated log P (*i.e.*, the digit of the whole period of the PRN series) and the number of PRN series with the power approximation curve. The digit of the whole period of the 10000 PRN series is log P = 54991 (*i.e.*, $P \approx 10^{54991}$); it is large enough for security applications in real life. The approximation curve shows the same tendency as ref. [2]; $\log P \approx 9.85n^{0.9263}$.

The whole period corresponding to *Method-3* in Table 4 is estimated by using the approximation curve. The digit of the whole period of the 33554432 PRN series is estimated as $\log P \approx 1.06 \times 10^8$, that is, the whole period is estimated at $P \approx 10^{100000000}$. It is huge for today's security applications.

According to the IBM road map of practical quantum computing announced on May 10th, 2022, IBM's 2025 goal is a 4,000+ qubit processor built with multiple clusters of modularly scaled processors [13]. It suggests that the quantum computer system is expected to compute 2^{4000} ($\approx 10^{1204}$) operations at once until 2025, therefore, the huge period in this work may be useful in the future.

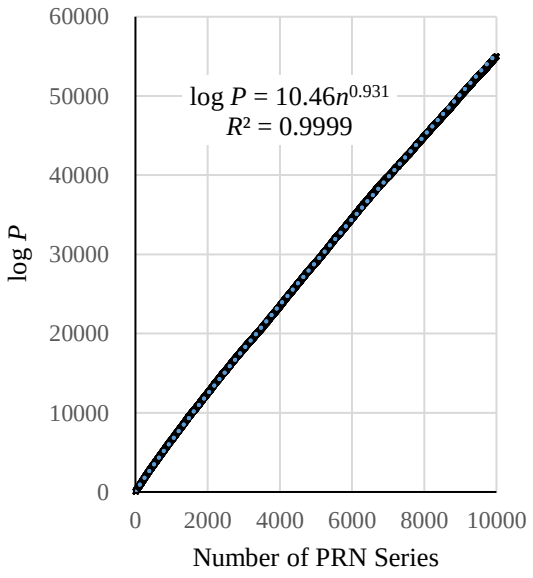

Figure 4: The digit of the whole period of CRNNs (log P) and the number of PRN series.

The parameters of the power approximation are shown in Table 5, for not only mixed subseries (α and β) but also for each subseries (α or β).

Table 5: Parameters of Power Approximation Curve. [a)]

Subseries	A	B	R^2
α	10.07	0.9351	0.9999
β	10.32	0.9338	0.9999
α and β	10.46	0.9310	0.9999

a) The parameters of the approximation curve: $\log P = A \times n^B$.

3. Conclusion

The novel iteration method with the modified activation function has been proposed to implement the secure and fast random number generator.

The modified activation function (*Method-3*) has accelerated the pseudo-random-number generation to 56.8 Tb/s by using NVIDIA GPU (A100) and CUDA 11.6. The CUDA context of the pseudo-random number generator (PRNG) has consisted of 1.67×10^7 threads (1.67×10^7 CRNNs and 3.34×10^7 subseries). The whole period of the PRNG is estimated at $P \approx 10^{100000000}$ ($\log P \approx 1.06 \times 10^8$) by the approximation curve based on the L.C.M. of the 10000 PRN series.

The rate of the generation with modified activation function (*Method-2*) in our precedent report also has been measured in this work as a faster rate, 57.6 Tb/s.

4. Future Work

Ultra-high-speed PRNG has successfully been implemented by the new activation function and the latest GPU, NVIDIA A100, which is equipped with 432 Tensor Cores. Although the Tensor Cores accelerate matrix operation, in this work the Tensor Cores have hardly been used so far. Making the best use of A100 is the most important aim in the future, and the key technology may be an effective and practical use of the Tensor Cores, and probably effective data migration between host and device memories.

Method-2 and *Method-3* are just examples of a new type of activation function, therefore the optimization is still underway; for better results for the NIST SP800-22 test, faster PRNG, and also predictability of the PRN series.

In the future, the proposed method will be applied to the secure CAN, the Autonomous GPU Accelerated system (AGX), and the security system on GPUs, *e.g.*, security application for IoT edge devices because the appropriate scale can be selected by using the equation in Figure 4.

Acknowledgments

Part of the experimental results in this research was obtained using supercomputing resources at Cyberscience Center, Tohoku University. Special thanks to the staff members of Iwate University Super-Computing and Information Sciences Center.

References

[1] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: a review," *IEEE Trans. Intell. Transport. Syst*., vol. 16, no. 2, pp. 534-545, 2015.

[2] H. Yoshida, Y. Akatsuka and T. Murakami, "Implementation of High-Performance Pseudo-Random Number Generator by Chaos Neural Networks using Fix-Point Arithmetic with Perturbation," *Proceedings of Papers, NOLTA 2018*, pp46-49, 2018.

[3] H. Yoshida, H. Fukuchi and T. Murakami, "Implementation of High-Speed Pseudo-Random-Number Generator with Chaotic and Random Neural Networks," *Proceedings of Papers, HICSS53 2020*, pp.6418-6425, 2020.

[4] H. Yoshida and T. Murakami, "Non-Attractive Periodic Trajectory Formation Mechanism on Random and Chaotic Time Series," Knowledge Innovation Through Intelli-gent Software Methodologies, Tools and Techniques, H. Fujita et al. (Eds.), IOS Press, 327, pp.197-208, 2020.

[5] H. Yoshida and T. Murakami, "Non-Attractive Periodic Trajectory Formation Mechanism on Random and Chaotic Time Series PART II," *Proceedings of Papers, NOLTA 2020*, pp.25-28, 2020.

[6] Z. Liu, T. Murakami, S. Kawamura, and H. Yoshida, "Fast Stream Cipher based Chaos Neural Network for Data Security in CAN Bus," *Advances in Science, Technology and Engineering Systems Journal*, 5:5, pp.63-68, 2020.

[7] H. Yoshida, Y. Kon, and T. Murakami, "Chaos Neural Network for Ultra-Long Period Pseudo-Random Number Generator," *Proceedings of Papers, ITISE 2017*, vol.1, pp.102-113, 2017.

[8] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, and J. Vo. S. Dray, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST SP800-22 rev.1a, Revised: July 2015 (sts-2.1.1)," Lawrence E. Bassham III, 2015.

[9] H. Yoshida, T. Murakami, and S. Kawamura, "Study on Testing for Randomness of Pseudo-Random Number Sequence with NIST SP800-22 rev. la," *Technical Reports of IEICE*, vol.110, pp.13-18, 2012.

[10] H. Yoshida and T. Murakami, "Activation Functions for Chaotic and Random Neural Networks," *Proceeding of Papers, JSST2022*, pp.351-354, 2022.

[11] NVIDIA DEVELOPER, "NVIDIA CUDA ZONE", https://developer.nvidia.com/cuda-zone (accessed-2022-06-01).

[12] Amazon Web Service, "Amazon Corretto Documentation" https://docs.aws.amazon.com/corretto/index.html (accessed-2022-06-01).

[13] IBM, "IBM Unveils New Roadmap to Practical Quantum Computing Era; Plans to Deliver 4,000+ Qubit System" https://newsroom.ibm.com/ (accessed-2022-06-01).