A General Pseudo-Random Number Generator Based on Chaos

Jianwen Lv, Xiaodong Li, Tao Yang, Haoyang Yu, and Beisheng Liu

Abstract In view of the shortcomings and problems of the commonly used C language pseudo-random number generation function, the existing C language pseudo-random number generation function is improved, a general pseudo-random number generator based on chaos is proposed, and multiple types of random number acquisition interface are introduced. While improving the randomness of the original C language pseudo-random number function, the procedure of improvement also enhances its versatility and can better meet the needs of different types of random numbers. The test results of the pseudo-random number generator show that the random number generated by it has good randomness, and the function call is convenient and flexible.

Keywords Chaos · Pseudo-random number generator · Random sequence retrieval · Universality

1 Introduction

Pseudo-random number generator (PRNG) is widely used in various fields such as system simulation and security [1]. Based on a reliable and efficient pseudo-random number generator, the system's operation, evolution, and development process are truly described in the system simulation. In the field of information security based on cryptography, pseudo-random number generators also play an important role. Key generation, digital signatures, authentication and identification, and various secure communication protocols are inseparable from high-quality random numbers. In a

J. Lv · X. Li (✉) · H. Yu · B. Liu
Beijing Electronic Science and Technology Institute, Beijing, China

T. Yang (✉)
The Third Research Institute of Ministry of Public Security, Shanghai, China
e-mail: yangtao@stars.org.cn

S. Mu et al. (eds.), *4th EAI International Conference on Robotic Sensor Networks*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-70451-3_9

sense, the security of random numbers determines the security of the entire security system [2].

There are four commonly used pseudo-random number generation functions in the C language standard library <stdlib.h> – rand function, srand function, randomize function, and random function. But these functions have obvious defects and poor ease of use. This chapter analyzes the existing pseudo-random number generation function in C language, proposes a general pseudo-random number generator scheme based on chaos, and designs a more general function interface. Compared with the original random number generation algorithm in C language, the pseudo-random number generator has enhanced anti-cracking ability and can generate pseudo-random numbers of different lengths, different ranges, and different types according to the needs of users. In addition, a key string is generated after the pseudo-random sequence is generated, which can be used to retrieve the lost pseudo-random number sequence under special circumstances. The test results show that the method proposed in this chapter has good performance in terms of randomness, key sensitivity, and compatibility.

2 Analysis of Commonly Used Pseudo-Random Number Generating Functions

2.1 Rand Function

The principle of the rand function in generating a pseudo-random number is to read a number as the seed parameter of the function. This parameter is determined after the computer is turned on. The function performs initialization operations based on the seed parameter and generates a pseudo-random sequence by iteration.

2.2 Srand Function

The principle of the srand function in generating pseudo-random numbers is the same as the rand unction, however, to make up for the shortcoming that the rand function generates the same pseudo-random sequence every time it is turned on, the srand function provides the function of customizing seed parameters.

2.3 Randomize Function

The randomize function is a random number generator initialization function, which is equivalent to a function that can modify the system seed parameters. Therefore, as long as the randomize function is called before the rand function to modify the system seed parameters, it can also generate a non-repetitive pseudo-random number sequence. However, the highest accuracy of the system time it obtains is still seconds, which is also vulnerable to exhaustive cracking attacks.

2.4 Random Function

Although the random function is similar to the rand function in usage, it can limit the pseudo-random number generation range by setting the value of formal parameters [4]. This function limits the value of the pseudo-random sequence by taking the remainder of the seed parameter iteration, but its shortcomings are also very obvious, that is, the minimum value of the range of values cannot be limited and the minimum value can only be 0.

3 One-Dimensional Logistic Map

Due to the ergodic, random, and initial value and parameter sensitivity characteristics of chaotic maps, chaotic sequences generated using this property have good cryptographic properties [3].

The one-dimensional logistic map is a very simple chaotic map from the mathematical form. As early as the 1950s, many ecologists used this simple difference equation to describe the change of population [5]. This system has extremely complex dynamic behavior and is widely used in the field of secure communications. Its mathematical expression is as follows:

$$f(x) = \mu x\,(1 - x)\,, \quad x \in [0, 1] \tag{1}$$

Among them, $\mu \in [0,4]$ is called logistic parameter. Research shows that when $x \in [0,1]$, the logistic mapping function is in a chaotic state [6].

The chaotic system is extremely sensitive to the initial value, the difference between the initial value is very small, and the difference between the values after several iterations is extremely large. For example, when $\mu = 4$ and the initial value $x(0) = 0.3$, after 20 iterations, $x(20) = 0.3794$. When the initial values differ by only 0.00001, that is, $x(0) = 0.300001$, $x(20) = 0.0084$, and after 20 iterations, $x(20) = 0.0084$. The iteration result differs by 45 times. The results of the two iterations are shown in Figs. 1 and 2. The extremely sensitive nature of the initial value determines that the time series generated by the chaotic system is highly unpredictable [7].

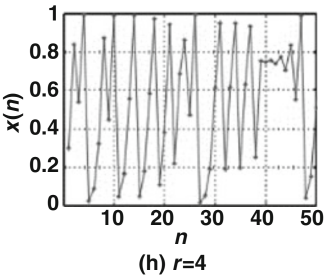

Fig. 1 *x* iteration result with initial value of 0.3

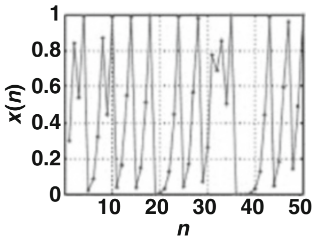

Fig. 2 *x* iteration result with initial value of 0.300001

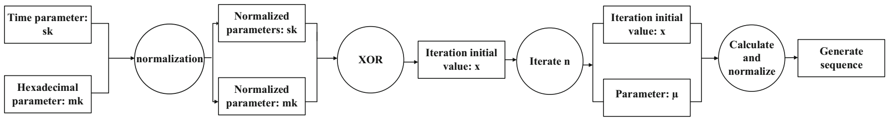

Fig. 3 Algorithm workflow

4 Algorithm Design Principles

For the two initial parameters μ and x of the traditional one-dimensional logistic mapping function, only when their values are within the specified interval, the sequence generated by the logistic mapping function is non-periodic and non-convergent. In addition, the generated sequence must converge to a certain value [5] (Fig. 3).

This chapter proposes not to directly generate the entire pseudo-random number sequence, but to generate a single pseudo-random number at a time through grouping iterations, etc., to effectively improve the randomness of the generated sequence. At the same time, the two parameters of this improved one-dimensional logistic mapping function can be determined by multiple factors. This system uses the results of four sets of parameter iterations as the coefficient μ of the system. The initial value *x* of the chaotic sequence is determined by three parameters mk, sk, and *n*, and they are defined as follows:

μ: This parameter is four sets of system parameters used for iteration, and its range is $(0 \leq \mu \leq 4)$.

sk: The parameter sk is obtained by multiplying the system parameters year, month, day, hour, minute, second, and microsecond in this scheme.

mk: The parameter mk is a set of four fixed hexadecimal parameters. Each time, one of the four sets of parameters is used to iterate in order to prevent the problem of randomness caused by using only one parameter.

n: The parameter n represents the number of system pre-iterations before generating chaotic sequences. By increasing the number of iterations, a more random chaotic sequence can be generated.

5 General Pseudo-Random Generation Interface Design

A good pseudo-random number generator needs to take into account both randomness and versatility. Randomness can be tested with NIST and other related testing software. At present, the randomness of most pseudo-random number generators can pass the test, but the versatility is not strong. In order to improve the versatility, this chapter proposes a set of pseudo-random number generator functions and provides an external interface to facilitate other programs to call. The required pseudo-random sequence can be obtained by entering the relevant parameters.

Different functions of the interface are implemented in different interface functions. This chapter provides three basic types of random number generation interfaces, which can generate corresponding integer pseudo-random numbers, floating-point pseudo-random numbers, and character pseudo-random sequences. The corresponding functions are: hdintrand function, hddoublerand function, hdcharrand function.

6 Performance Test

Regarding the testing of random/pseudorandom sequences, the NIST (National Institute of Standards and Technology) provides 16 detection indicators [8]. For each binary sequence, each test indicator will give a P-value as the test result. If the value is greater than a pre-set threshold α, it means that the randomness of the test sequence is 1-α, or the sequence has passed the randomness test of the detection index; otherwise, it means that the index has not passed test. In this paper, α is set to 0.01, that is, if the sequence passes the test, the credibility of its randomness is 99%.

At the same time, 100 pseudo-random sequences were generated using the algorithm and random module based on the work described in this chapter, and each sequence has a length of 100,000 bits. These sequences are applied in randomness tests, the P- value of each detection index is shown in Table 1. Based on randomness test, the results generated by the pseudo-random number generator in this paper are better than the random function test results overall. Based on the comprehensive experimental results, it can be considered that the pseudo-random number generator

Table 1 Randomness test results according to NIST indicators

Test item	P-value generated by our algorithm	Random number generated by random module
(Frequency)	0.017912	0.920402
(Block frequency) ($m = 128$)	0.107293	0.038105
(CumulativeSums) – forward	0.242986	0.855781
(CumulativeSums) – reverse	0.186566	0.929658
(Runs)	0.689019	0.362766
(LongestRunofOnes)	0.392456	0.237054
(Rank)	0.105618	0.329090
(FFT)	0.186566	0.912193
NonOverlappingTemplate	0.392456	0.531696
(OverlappingTemplate) ($m = 9$)	0.141256	0.136782
(Universal)	0.689019	0.439164
(ApproximateEntropy) ($m = 10$)	0.311542	0.826978
(RandomExcursions) ($x = +1$)	0.275709	0.152625
(RandomExcursionsVariant) ($x = -1$)	0.162606	0.496248
(Serial) ($m = 16$)	0.141256	0.161351
(LinearComplexity) ($M = 500$)	0.179129	0.088454

proposed in this chapter can generate pseudo-random number sequences with good cryptographic characteristics [8].

7 Conclusion

In this chapter, a C-language universal pseudo-random number generator based on chaotic system is designed. The algorithm of this pseudo-random number generator is described in detail, its versatility and security are introduced, and its chaotic characteristics are analyzed. At the same time, this chapter analyzes some defects of the existing pseudo-random number generation function in C language. The comparison test results show that the random number generator design scheme given in this chapter has good random performance and strong versatility. According to the random sequence and other characteristics, the pseudo-random sequence generated in this study is enough to meet the randomness requirements.

Acknowledgments This work was partially supported by the National Natural Science Foundation of China (grant number 62072014), the Beijing Natural Science Foundation (grant number L192040), the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (grant number VRLAB2019C03), and the Key Lab of Information Network Security, Ministry of Public Security (grant number C18612).

References

1. Liu, P.H., Lu, H.X., Gong, G.L., et al.: High-speed arbitrary distributed pseudo-random number generator based on FPGA[J]. J. Appl. Sci. **30**(3), 306–310 (2012) (in Chinese)
2. Su G.P.: Research on random sequence and application of wavelet analysis in information security[D]. Graduate School of Chinese Academy of Sciences (Institute of Electronics), 2002 (in Chinese)
3. Zhang, X.F., Fan, J.L.: A new piecewise nonlinear chaotic map and its performance[J]. Acta Phys. Sin. **59**(04), 2298–2304 (2010) (in Chinese)
4. Li, K.J.: On the random number problem in C language[J]. J. Changchun Univ. **6**, 64–68 (2008) (in Chinese)
5. Zhu, Y.P.: Performance comparison research of various chaotic systems[J]. Microcomput. Appl. **35**(12), 4–6 (2016) +9. (in Chinese)
6. Zhao, G., Tian, Y.L., Yin, S.: Progresses and perspectives of Chaotic Cipher algorithm[J]. J. Beijing Inst. Electron. Technol. **24**(04), 9–14 (2016) (in Chinese)
7. Sun, X.H., Lin, Q.H., He, Y.W.: Pseudorandom number generator based on combined chaotic map[J]. Chin. J. Sci. Instrum. (S1), 805–807 (2006) (in Chinese)
8. Zhang, Y.Q., Li, S.B., Qu, S., Lv, K.L., Liu, C., Xu, X.R.: NIST randomness test method and application[J]. Comput. Knowl. Technol. **10**(26), 6064–6066 (2014)