ICTA'07, April 12-14, Hammamet, Tunisia

A Modified Method for Generating Secure Accessibility Pseudorandom Bit Sequences

Dr. Abdulameer K .Hussain
Computer Science Department, Zarqa Private University, Jordan
ameerk53@yahoo.com

Abstract

This paper presents a modified algorithm for ANSIx9.17 version [1], which is used for generating pseudo-random number sequences. This algorithm depends on fixed execution steps and fixed variable choosing. Also it uses a fixed cryptographic algorithm, DES. The modified algorithm introduced more modifications include using variable parameters instead of fixed parameters and different cryptographic algorithms in a random selection done by the user to get more secure random generation., that enhances a secure accessibility .

Keywords

Randomness, Pseudo-Random Generators, Cryptography, Security, Agreement, Seed, and Accessibility.

1. Introduction

The security of many cryptographic systems depends upon the generation of unpredictable quantities [1].Security systems today are built on increasingly strong cryptographic algorithms that foil pattern analysis attempts. However, the security of these systems is dependent of generating secret quantities for passwords, cryptographic keys, and similar quantities. The use of pseudo-random processes to generate secret quantities can result in pseudo-security. The sophisticated attacker of these security systems may find it easier to reproduce the environment that produced the secret quantities, searching the resulting small set of possibilities, than to locate the quantities in the whole of the number space.

Choosing random quantities to foil a resourceful and motivated adversary is surprisingly difficult [2]. Although the term is appropriate and is used in the field, the phrase "random numbers" can be misleading. To many people, it suggests random number generator functions in the math libraries which come with one's compiler. Such generator functions are insecure and to be avoided for cryptographic purposes [3].

Random number generation is used in a wide variety of cryptographic operations, such as key generation and challenge/response protocols. A random number generator is a function that outputs a sequence of 0s and 1s such that at any point, the next bit cannot be predicted based on the previous bits. However, true random number generation is difficult to do on a computer, since computers are deterministic devices. Thus, if the same random generator is run twice, identical results are received. True random number generators are in use, but they can be difficult to build.

Pseudo-random number generators are often based on cryptographic functions like block ciphers or stream ciphers. For instance, iterated DES encryption starting with a 56-bit seed produces a pseudo-random sequence [4].

Randomness and random numbers have traditionally been used for a variety of purposes, such as dice games. With the advent of computers, people recognized the need for a means of introducing randomness into a computer program. Surprising as it may seem, however, it is difficult to get a computer to do something by chance. As the name suggests, pseudo-random numbers are not truly random. Rather they are computed through a mathematical formula or simply taken from a precalculated list. Random numbers are used for computer games as well as used on amore serious scale for the generation of cryptographic keys. Furthermore, random generators are used to facilitate access for authorized user only and thus they are the best tools for protecting data against illegal access [5].

2. Background and classification

A random bit generator is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits .this generator can be used to generate (uniformly distributed) random numbers. For example, a

random integer in the interval [0, n] can be obtained by generating a random bit sequence of length $\lfloor \lg n \rfloor + 1$ and converting it to an integer; if the resulting integer exceeds n, one option is to discard it and generate a new random bit sequence.

Basically, most random number generators work like this:

1. A seed value is obtained in some way, usually by user input, or checking the computer's clock cycle.
2. This seed value is sent through a formula that will calculate a new random number.
3. This new random number is used as the seed value for the next random number.

A Pseudo Random Number Generator (PRNG) suitable for cryptographic applications is called a cryptographic secure PRNG (CSPRNG) which is cryptographically stronger than PRNG [6]. Random numbers can be classified either true-random or pseudo-random. The later are algorithms which imitate the former [3].

Information collected for the seed does not need to be truly random, but unguessable and unpredictable. Some PRNGs, that are software based, use hardware based PRNGs for seed generation [7]. In order to use a random number generator on most computers, you should first "seed" the sequence [8].

3. Security, accessibility and PRNG

3.1 Accessibility

Security and accessibility include the mechanisms of authentication and authorization. Authentication means that the program assures that only users with defined roles, such as administrators and students, can access the system. Authorization assigns appropriate privileges, such as access to information, write or delete privileges, to users once they are authenticated. We must consider the following :

- Are records accessible to the instructor?
- Can data be easily exported for analysis?
- Does access to records meet legal and confidentiality requirements?
- Is summary data anonymous?
- Can users access their own records?
- Is the data safe from tampering? [9]

3.2 Uses of pseudo-random generators

Many fields of research (e.g. physics and even finance [10]) are increasingly relying on large computer simulations to study phenomenons which can not be observed directly for various reasons. In these circumstances, there are few, if any, alternate validation methodology to avoid an unforeseen subtle statistical interaction between the generator and the properties of the simulated phenomenom [11].

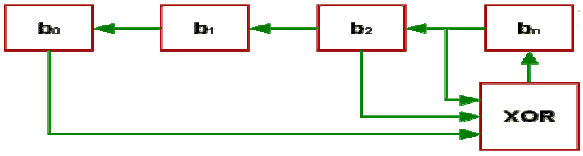
The PRNG deals with some of fields of science, such as, mathematics, computer science, physics, and cryptography [12, 13].

3.3 Traditional pseudo –random sequences

A typical pseudo-random number generation technique, known as a linear congruence pseudo-random number generator, is modular arithmetic where the N+1th value is calculated from the Nth value by

$$v_{n+1} = (v_n \times a + b)(\mathrm{mod}\ c)$$

The above technique has a strong relationship to linear shift register pseudo-random number generators, which are well understood cryptographically. In such generators bits are introduced at one end of a shift register as the Exclusive Or (binary sum without carry) of bits from selected fixed taps into the register. For example:



$$v_{n+1} = (v_n \times 2) + b_0 \ .\mathrm{xor}.\ b_2 \ldots)(\mathrm{mod}2\hat{}n)$$

The goodness of traditional pseudo-random number generator algorithms is measured by statistical tests on such sequences. Carefully chosen values of the initial V and a, b, and c or the placement of shift register tap in the above simple processes can produce excellent statistics [6].

4. Proposed Algorithm for pseudorandom generation

The basic ANSIx9.17 generator algorithm uses a fixed seed s of length 64-bit and an encryption key of DES as input to the generator. We modify these parameters to be an n-bit variable with another variable parameter m. Also two non DES keys to be two keys k1 and k2 are used instead of known fixed key. To obtain more secure random bit

generation, it is best to modify the original algorithm in order to double encryption instead of single encryption. To perform this task we can use different encryption algorithms instead of a fixed encryption algorithm. Finally, according to these modifications we can produce a large number sequences which may be twice the number obtained in the original algorithm,

The modified Algorithm

Input: A random (secret) n-bit seed s, a variable integer m, and two keys k1, k2, each of n-bit.

Output : m pseudorandom n-bit strings x1,x2,…….,xm .

1: Compute the integers I1 = Ek1(D), I2=Ek1(D), where D is the n-bit representation of the current date/time.

2: For i=1 to m

2.1 xi=Ek1(I1 XOR s)

xii = Ek2(I2 XOR s)

2.2 s1= Ek1(x1 XOR I1)

s2=Ek2(xii XOR I2)

3: Return x1,x11,x2,x22,x3,x33,……..xm ,xmn

5. Results

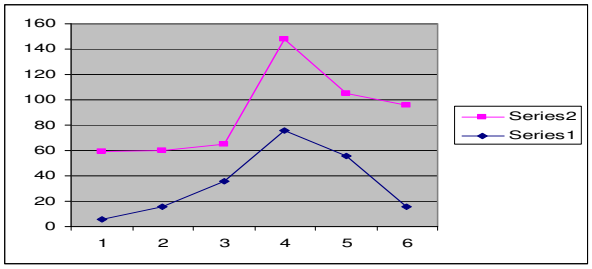


Figure 1. Randomness with m1 and d1

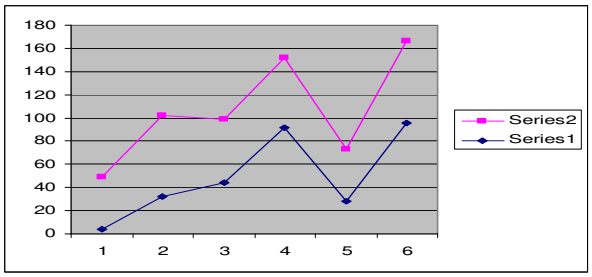


Figure 2. Randomness with m2 and d2

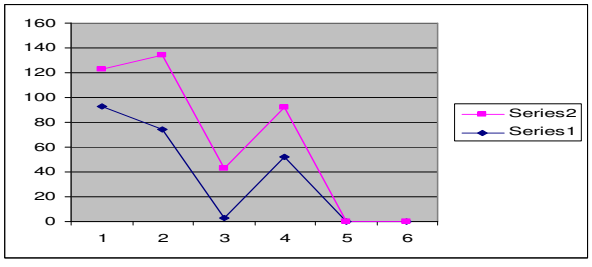


Figure 3. Randomness with m3 and d3

6. Conclusion

The modified method of pseudorandom generator presented in this work relies on applying several modification that were used in the original ANSI X9.7 The first modification is using variable seed value to deceive the eavesdropper because applying fixed seed (although it is secret) will take some time to discover this secret according to the exhaustive search, So, by applying variable seed the eavesdropper will take much more additional time to discover this seed making the modified algorithm more secure and enhance authorized access to multiple data. The second modification is also using a variable integer besides the variable seed value .The third and most important modification is applying double encryption with two different keys. The double encryption makes the system more secure. The algorithms used in double encryption are different and they are changed either periodically or by the agreement of the parties. The output of the modified algorithm is twice the output obtained by the original algorithm .This leads to produce efficient random sequences because adding more bits to the fixed output of each algorithm affects the efficiency of that random bit sequences.

7. References

[1] J. Menezes, P. C. van Oorschot, and S. A. Vanstone, “Handbook of Applied Cryptography “, CRC Press, 1997.

[2] www.packetizer.com

[3] world.std.com

[4] www .rsasecurity.com

[5] D. Eastlake, Network Working Group, 3rd, MIT, December 1994

[6] Donald Knuth., "The Art of Computer Programming ", Vol 2 : Seminumerical Algorithms, Third Edition . " Addison-Wesely ", 1997.

[7] www.ee.oulu.fi

[8] cs.gmu.edu

[9] www.dental.pitt.edu

[10] Brotherton-Ratcliffe, Rupert, Using Quasi-Random Sequences in Monte-Carlo Valuation of Path-Dependent Options, Risk Magazine, December 1994, and also in Canadian Treasurer, vol 11, number 2, April 1995, pp 36-38.

[11] Vattulainen, I., Ala-Nissila, T., and Kankaala, K., Physical Tests for Random Numbers in Simulation, Physical Review Letters, Vol. 73, Number 19, 7 November 1994, pp. 2513-2516

[12] Micali, Sylvio, and Schnorr, Claus P., *Efficient, Perfect Polynomial Random Number Generators*, Journal of Cryptology, Vol 3 (1991), pp 157-172

[13] www .connotech.com