The International Conference on Research Perspectives : IoT in Hybrid Grid Integrated Renewable Energy Sources
In association with International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)

Random Number Generator Using Sound

Abhishek Francis S[1], Dr. Pamela Vanitha Eric[2]
[1]MTech Student, Department of CSE, NHCE, Banglore, Karnataka, India
[2]Professor, Department of CSE, NHCE, Banglore, Karnataka, India

ABSTRACT
True random number generators can significantly contribute to the development of high security cryptographic schemes, such as those required for use in military applications. This article presents some the results of an innovative method for the generation of truly random number sequences, based on environmental noise measurements.
The statistical properties of different noise types have been studied. Based on this study, an efficient random number generator has been developed that uses signals from the built-in microphone that is ubiquitous most current personal computers and other personal information processing systems. Statistical measures have been determined that measure the randomness qualities of the output sequence. These measures have been studied for different input noise properties.
Key-Words: - True Random Number Generation, Cryptography, Sound

I. INTRODUCTION

From simple gambling thousands of years ago, to modern statistical sampling, humans have created random number generators. Methods ranging from a simple coin flip to complex deterministic equations have been employed for amusement, statistics, and science. However, the use of random number generators, even today, is still not completely understood. Our foray into random number generation barely scrapes the surface of the subject. Thus, much of this work is a compilation of previous research. Our aim has been to condense the wealth of information on random number generation into a simple overview and further, execute several established generators. Through original computer applets, we attempted to test the degree of randomness of the generators, using several traditional statistical tests. Computer systems and telecommunications play an important role in modern world technology.
The communication and data transfer through computers touch almost every aspect of life, i.e., transferring data, tracking personal data, trading over the internet, online banking and sending emails. As more vital information is transferred through wire or wireless means, the need to safeguard all this data from hackers is growing. All these security concerns emphasize the importance of developing methods and technology for the transformation of data to hide its information content, prevent its modification, and prevent unauthorized use.
Random number generation is a fundamental process for protecting the privacy of electronic communications. It is a key component of the encryption process that protects information

from attackers by making it unreadable without the proper decryption processes. Since the strength of an encryption mechanism is directly related to the randomness of the binary numbers used in it, there has been an enormous need to design and develop an efficient random number generator that can produce true random numbers to implement a safe and secure cryptographic system

II. RANDOM NUMBER GENERATORS

As one can imagine, random number generation is quite a broad topic. Involves producing a series of numbers with no recognizable patterns or regularities, that is, appearing random." Thus, computers do not hold a monopoly on random number generation. Dice rolls, coin flips, drawing lots, and other forms of physical means can indeed produce

streams of numbers that one could not predict.
These were the earliest random number generators. As larger and larger streams of numbers were required for scientific procedures, physical random number generators lost value; scientists required the streams within a reasonable time-frame, with little effort. With the advent of computing, random number generation received a needed modern boost.
A computer is capable of two distinct types of random number generation. The first is comparable to a physical random number generator. What is known as a hardware random generator can extract information from a "weakly" random physical process, and convert the information into a stream of numbers. Processes used include thermal noise, nuclear decay, or "quantum" processes, such as the agitation of electrons.
Computers also excel at iterating deterministic equations. Generators based on a deterministic, iterated function are called pseudorandom number genera- tors. These generators are the primary ones used in scientific research in our modern world. These three distinct genres of generators offer unique ways of random number generation. In studying these generators, one can learn much about randomness, and its applications in science.

III. WHAT IS RANDOM?

To see the value in random number generation, one needs to know what is meant by a random number. For accuracy, we refer to Edwin Lindauer, in his "Methods of Random Number Generation," [5] to define the term. "A random number is defined to be a series of digits (a) where each digit has the same probability of occurring and (b) adjacent digits are completely independent of one another (if we know one digit, we still have no way of predicting the next one)." That is, given any number of digits in a stream of digits, we could not predict the following, or previous numbers, for example.
This definition is, of course, not all-encompassing. Many degrees of randomness in streams of "random" numbers exist. In any deterministic generator, the streams of numbers created cannot be called "random" since they are created through an algorithm. Thus, they are known as "pseudo-random" numbers. Up to the discretion of the tester, a random number generator may be designated "statistically random" if it passes certain chosen tests. For many purposes, "statistical randomness" is acceptable, though such "random" streams are not truly random, and may even fail to "appear" random if further testing is employed. However, one can estimate the degree of statistical randomness of a generator.

IV. MODERN USES OF RANDOM NUMBERS

One of the most famous uses for random numbers are those in statistical sampling.

Many facets of statistics require random numbers, such as choosing a representative sample and randomization of steps of an experiment to hide the protocol from the subject. It is also useful in creating the design itself. Certain types of statistical analysis require random number generators. In any of these uses, a faulty random number generator can nullify any results. Simulation of statistical events such as a coin toss, or computer simulation of physical phenomenon also requires random number generators.

Cryptography is another interesting application of random number generation. For the utmost security, the streams of numbers used must not only be random, but also from an unknown, that is, new source. For example, the digits of π can provide random streams. According to Weinstein [6], the decimal digits of π, are thought to be normal, that is, the distribution of its decimal digits (0–9) has a uniform probability of 1/10 for each digit. There has been as of yet no widely accepted proof that any non-artificially constructed numbers are normal. However, the first 30, 000, 000 decimal digits of π have been shown to be uniformly distributed (Bailey, [1]). One might think that drawing from a possibly uniform distribution, like the digits of π, would be an efficient method of generation. However, well known streams like that of π, or commonly used deterministic generators are not very secure options.

Finally, Monte Carlo methods of simulation are yet another remarkable application of random numbers. This is a broad heading under which fall many algorithms which, instead of using a deterministic function, use random sampling to compute results. Weisstein [7] defines a Monte Carlo method as “any method which solves a problem by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties”. A common use of the Monte Carlo method is known as Monte Carlo integration. To simplify integration of a function with a complex domain, D, Monte Carlo integration randomly plots “many” points over a larger, simpler domain; each point can then be categorized: it is either inside the bounds of D, or outside. An estimation of area can then be made using the ratio of points in D to those outside of D but within the much simpler domain ([7]). Before this method came about, simulations were tested against already solved deterministic problems to test their success. In turn, Monte Carlo methods solve deterministic problems using a simulation. They are usually applied to problems with many variables and parameters, that is, a system too complex to solve deterministically. They could therefore be applied to dynamical systems; though not necessarily arising from complex multivariate equations, these systems are often difficult to examine deterministically. Monte Carlo methods have, in fact, been applied to fluid flow.

V. PHYSICAL METHODS HISTORICAL METHODS

In a sense, the earliest methods of random number generation were highly successful. Physical processes are not entirely random, but are usually so complex that they are unpredictable to the casual observer.

In the ancient world, chance was linked to fate. Those interested in their fate could throw dice in the hopes of divining their future. Primitive “generators” included the scattering of rice and the interpretation of the cracks of a turtle’s shell, once heated. These methods were not number generators, as they did not produce numbers; they required interpretation before recording. Actual random number generators were used as well. Specifically, in the I Ching, an ancient Chinese text, there are many methods discussed in depth of random number generation. These include coin-flipping and the counting of yarrow stalks. The random data produced by these methods was then used to divine fate.

These methods of divination morphed into games of chance, the first being simple bets on the rolls of a die. Games of chance date back to 2000 B.C.

VI. GENERATING NUMBERS FROM THE ENVIRONMENT

In our modern world, using physical processes to generate random streams may seem outdated. A deterministic random number generator, like those discussed in the next section, can speed one's work along substantially. However, there are instances in which such a generator is not useful. This is true in certain facets of cryptography. In cryptography, random numbers are used to generate keys for secure communication. If the numbers used are not random enough, security may be compromised.

In cryptography and other areas, ultra-secure options include generators called hardware random number generators [5].

These are based on physical processes like the historical generators mentioned. However, the processes used for the hardware generator are favored to be those with quantum quality, that is, quantum mechanics predicts these phenomena are fundamentally random. These include shot noise, which is "noise" created when photons from a lamp are directed to a photodiode. Noise occurs when the photons affect the circuit. Other phenomena are used which are not predicted random by quantum mechanics; these include thermal noise: the agitation of electrons inside a conductor. Quantum processes are hard to detect, as they are so small, thus other sources of noise are used, though these are more easily attacked. From the physical noise, a transducer converts the physical phenomenon into an electrical signal. Then, an amplifier is usually required to increase the amplitude of the signal to the macroscopic level, so it can be read by an analog to digital converter that finally yields a number. While generally very hard to predict, there are often issues of statistical randomness in the final data. One example is called bias. The data will often have a mean not equal to the expected 0.5 of a uniform distribution on the interval [0, 1]. Thus, most hardware generators have what is called a corrector

to bring the mean to the desired value. Another problem can arise if data is sampled too quickly. In such a scenario, short-term dependencies of physical phenomenon may cause the data to be correlated. Finally, hardware generators are sensitive to outside noise. This can effectually ruin a data set. Roger Davies, in his Hardware Random Number Generators, presents two spectral analyses of hardware random number generators, one that passes the significance test, and one that is badly contaminated by an outside frequency, and thus fails miserably (See Figure 1 and Figure 2 below). Note the difference between the distribution of the data between the horizontal lines in each graph.

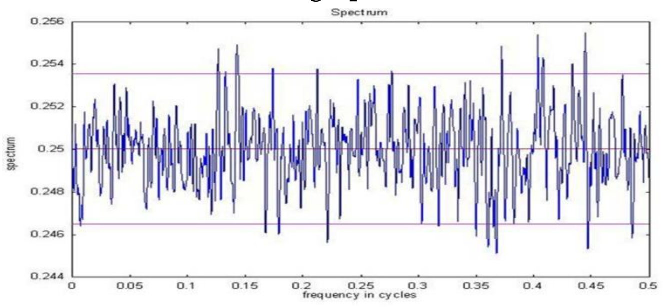

Figure 1 Spectral analyses of hardware random number generators within the frequency

Though these generators are often slow, the field is still expanding. A recent development is the use of lasers as a source of random numbers. Especially exciting in the new study, multiple sources were used, thus multiple streams of data were generated concurrently. Setups like this one could help speed hardware generators considerably in the future (Wu, [4]).

VII. SOFTWARE RANDOM NUMBER GENERATORS

Casting lots or using environmental white noise to generate random numbers is not always possible, or feasible. In many modern scenarios, casting lots or cracking turtle shells simply takes too much time. In a database where user information is encrypted with random prime numbers, rolling dice to determine numbers are impractical, to say the least, when dealing with thousands of users. Generating the numbers from the environment often takes time as well as dedicated computer hardware

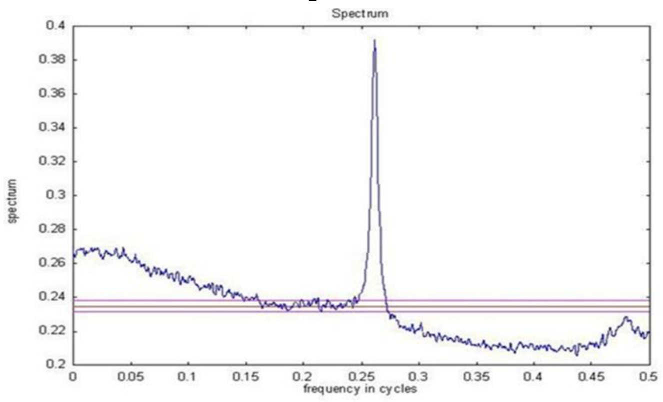

Figure 2 Spectral analyses of hardware random number generators outside the frequency

The specialized computer hardware makes these generators impractical for most users. Software random number generators allow for the creation of high-quality random number sequences quickly and without any specialized computer hardware.

The algorithms used in software random number generators are easily implemented on computers for use in cell phones, laptops, servers, and data centers. These algorithms have multiple parameters that dictate how the generator will perform. In most cases, these parameters are preset values except for a seed value. The seed value is typically time based and is used as a starting place for the production of the sequence. Unfortunately, these algorithms all have a cyclic period after which they will repeat. There are guidelines available for each generator to maximize its period. Sometimes the restrictions make selecting the proper initialization for a generator a difficult process. However, modern programming languages come with these random number generators built-in so that software engineers always have access to good random number sequences. There are many different algorithms for generating a sequence of pseudorandom numbers.

VIII. LINEAR CONGRUENTIAL GENERATOR

The most common category of software random number generators is the Linear Congruential Generators (LCG). An LCG is defined as the recurrence relation of the form:

Xn+1 = (aXn + c)(mod m), $n \geq 0$ m, the modulus; $m > 0$

a, the multiplier; $0 \leq a < m$ c, the increment; $0 \leq c < m$ X0, the seed value; $0 \leq X0 < m$.

According to Knuth [2][3], LCG's are known to have a period of length m if three conditions are met. First, c and m must be relatively prime. Second, a should be divisible by all prime factors of m. Third, a should be a multiple of 4 if m is a multiple of 4. In addition, the choice of m is typically the word size of the computer (usually 232 or 264) because that makes the binary modular operation very fast. Generator's whose parameters do not meet these requirements have significantly shorter periods and are therefore not suitable for many scenarios. LCG's excel in applications where a small memory footprint is crucial, such as in embedded systems like an internet router or car computer, because they need very little memory in order to execute.

IX. TRUE RANDOM NUMBER GENERATOR (TRNG)

A hardware true random number generator is an electronic device that generates truly random and unpredictable binary numbers. The output pattern of a TRNG is arbitrary and non-deterministic in nature, meaning that the output binary numbers cannot be reproduced even if internal design and seed of the generator is known. As complete unpredictability is the key aspect of the true random number generator, a seed given to the TRNG must be random. Fortunately, it is not so difficult to collect true unpredictable randomness by tapping a chaotic world. Some of the examples of physical random sources are, thermal noise, shot noise, atmospheric noise, radioactive decay and clock jitter. A TRNG can be fed a seed from such a physical random process to get true random outputs. Another characteristic of TRNGs is that they are non-periodic in nature (as opposed to PRNGs), meaning that output binary pattern of a TRNGs is never repeated even if the same seed is applied to the generator.

Chip manufacturing companies such as Intel are including random number generators in their chips. This paper illustrates a hardware-based Intel Random Number Generator for use in cryptographic applications.

The Intel hardware random number generators are based on unpredictable analog property such as junction or thermal noise. In this design, Intel RNG samples the thermal noise of

undriven resistors by amplifying the voltage across it. One significant problem associated with this noise amplification technique is that random components are associated with local pseudorandom noise sources such as temperature and power supply fluctuations. The effects of these sources are minimized by subtracting the signals sampled from two adjacent resistors.

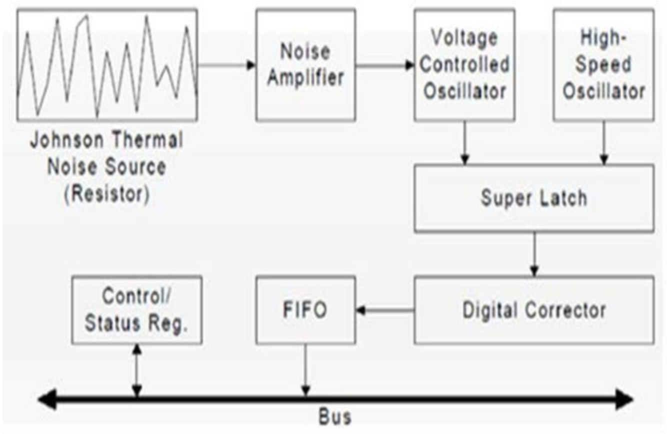

Figure 3 Thermal noise voltage amplifier for TRNG

X. SUMMARY OF CHARACTERISTICS OF RANDOM NUMBER GENERATORS

Below is the summary of the properties of a pseudo random and a true number generator.

Characteristic	PRNG	TRNG
Non-deterministic	No	Yes
Unpredictable	No	Yes
Reproducible	Yes	No
Periodic	Yes	No

Table 1 Properties of PRNG and TRNG

XI. CONCLUSION

Random number generation is a veiled subject; often used in mathematical settings, but not much studied. Our foray into number generators yielded an implementation of several software generators that seem to work well given certain seed values and parameters. Other choices of these values failed spectacularly. Given our limited knowledge on the barrage of tests available to test these generators, we cannot say definitively whether any of these generators would continue to perform, under more extensive testing. What we can report is the complexity of random number generation. A deeper understanding of such methods is required as we rely increasingly on random number generation globally.

XII.REFERENCES

[1]. D. H. Bailey, The Computation of π to 29, 360, 000 Decimal Digit using Borwein's' Quartically Convergent Algorithm, Math. Comput. 50 (1988) 283–296

[2]. Donald E. Knuth, Generating Uniform Random Numbers, Seminumerical Algorithms, 2nd ed., Reading, MA: Addison-Wesley Pub. 2 (1981) 9–38. Print. The Art of Computer Programming.

[3]. Donald E. Knuth, The Spectral Test, Seminumerical Algorithms. 2nd ed., Vol. 2, Reading, MA: Addison-Wesley Pub. 2 (1981) 89–114. Print. The Art of Computer Programming.

[4]. J. G. Wu, X. Tang, Z. M. Wu, G. Q Xia, and G. Y Feng, Parallel generation of 10 gbits/s physical random number streams using chaotic semiconductor lasers, Laser Physics, 22(10)(2012)1476–1480.Retrievedfrom http://link.springer.com/article/10.1134%2FS1054660X1210 0246

[5]. R. Davies, Hardware random number generators, 15th Australian Statis- tics Conference, 2000. Retrieved from http://www.robertnz.net/hwrng.htm

[6]. Eric W. Weisstein, Normal Number, From MathWorld–A_Wolfram_WebResource. http://mathworld.wolfram.com/NormalNumber.html

[7]. Eric W. Weisstein Monte Carlo Integration, From MathWorld–A_Wolfram_Web_Resource. http://mathworld.wolfram.com/ MonteCarloIntegration.html