Advanced Telecommunication's Exam

Random Number Generation and Entropy Techniques

Gregory Partridge

***Abstract*—The aim of this essay is to explore Random Number Generators and their associated entropy techniques. The two types of techniques used are True Random Number Generators(TRNG) and Pseudo Random Number Generators(PRNG). TRNGs are more secure than PRNGs, hence are widely used where security is very important, like cryptography. PRNGs is used more on less security dependent applications such as media or simulations. This essay describes the theoretical basis of entropy techniques up to and including the most recent developments.**

I. Introduction

A. What is a Random Number Generator

A Random Number Generator (RNG) in its simplest definition is the process of generating a sequence of numbers where the succeeding digit in a sequence is not able to be reasonably predicted. To do this there have been a plethora of distinct entropy techniques which have been adopted for multiple different functions which will be explored thoroughly throughout this essay. These being the type of random numbers, the real world application of random number, entropy techniques employed, security and performance and the future of RNG.

B. What constitutes an Entropy Technique

Entropy is a method for creating randomness in a system. When entropy is applied to computing, then randomness is obtained from the operating system or from the application of a given system. There are many different methods for doing this, but these methods can be split broadly into two distinct groups called True Random Number Generators and Pseudo Random Number Generators. These two differ in the way in they generate randomness.

II. True Random numbers and Pseudo Random Numbers

A. True Random Numbers

True Random Numbers are numbers generated by the means of a physical process rather than an algorithmic approach. True Random Numbers usually contain three characteristics: unpredictability, uniform distribution and lack of patterns. They are used a lot in cryptographic applications where it is important that 'attacks' or infiltrations are not able to figure out or decipher possible patterns and thus compromise the security of a system. It should be impossible to uncover a code and the safest way to deal with this is to use True Random Numbers. The drawback of this is that generating such codes is time consuming and expensive to produce due to the nature of how they are generated which is physical means as well as that they are tied with physical process for speed.

1) Pseudo Random Numbers: Pseudo Random Numbers are produced primarily through the use of algorithms. The standard way is for Pseudo Random Number generators is to start with a seed state. Due to this they are unsuitable for cryptographic applications due to as their repeating nature. The reason we use them is for their efficiency as well as their application in simulation alongside modelling applications. These simulations can be repeated if the seed is known if there is a need to replicate the result.

B. How it works

True Random Number Generators (TRNGs) and Pseudo Random Number Generators (PRNGs) work on very different principles. True Random Numbers work by taking the source of its randomness from a physical source when compared to the algorithmic approach of Pseudo Random Numbers. Such physical sources being, but not limited to, quantum random properties such as nuclear decay or the fluctuations in vacuum energy as well as thermal phenomena such as the thermal noise received from the likes of a resistor or atmospheric noise. These methods will be expanded upon with significantly more depth later.

Pseudo Random numbers are calculated through an algorithmic approach. The main correlating factor between all Pseudo Random Number approaches is that they have a seed key and are all expanded upon from here. A simple example of one, though one that is not so much in use these days, is the Linear Congruential Generator:

$$X_{n+1} = (aX_n + c) mod m$$

where m is the modulus, a is the multiplier, c is the increment and X0 is the seed. No matter how complicated you make the algorithm, if the seed becomes known it is possible to calculate past and previous values rendering the security to Pseudo Random Number Generators very poor.

III. Real World Applications

How does this apply to the real world? In this section we will be going over the application and the necessary qualities needed to thrive in the certain given fields.

A. Gambling

Gambling is at its heart a luck-based game. Though there is skill involved a lot of that is working around that randomness element that is unavoidable. In online gambling sites the need for a Random Number Generator is unavoidable. The main way of obtaining the randomness for these applications or websites is through the use of pseudo random number generators. These are used in many games on many sites, as they are essential in the simulation of dice rolls, card draws as well as the results of slot machines.

For dice and cards it is quite easily understood how this would play out. Each of the random numbers corresponds to an outcome for either the die or one of the cards from the deck or decks of cards. For the slot machine it takes a similar approach but not quite as intuitive. The way slot machines work is that a value is assigned to each symbol in the reel and if for example there were six symbols and three reels, then the Random Number Generator would generate a value between one and six three times for each of the reels.

B. Statistical Sampling

Statistical Sampling is very dependent on Random Generators when selecting the samples from the sample pool. The process is where each item in the population has an equal chance of being chosen. The way Random Number Generators come into use here is their use in creating Random Number Tables. These are used as they are good at ensuring that all members of a population have an equal chance of being selected. It also allows for selecting specific sub groups of a population depending on the corresponding value they have in the Random Number Table that allows for selecting a sample of a sub demographic from a general population. Statistical programs, like SPSS, R and Stata, have random number generators embedded as part of the program for use in these applications.

C. Computer Simulation

Simulations that use a stochastic approach are reliant on Random Number Generators for the random occurrences. A stochastic approach refers to being able to be analysed statistically though is not able to be predicted precisely. These methods commonly adopt Pseudo Random Number approach primarily to take advantage of the seed aspect of the technique. If they seed is saved it is possible to reuse it in order to repeat the simulation to gather the same result as the random numbers wouldn't necessarily be random but would actually be the same ones as before causing the same result. Though Pseudo Random Numbers are the more commonly used, True Random Numbers also have place in being used if a true random is needed. The Random Number Generator chosen really comes down to what is best suited for the simulation.

An example of the application of a Random Number Generator to a simulation would be how random numbers were used to help improve simulations of stochastic models of neural cells [5]. The process of simulating the biological process of the electric potential flow of the neural cell's membrane has a lot of demand on random variables. They used both a Pseudo Random Algorithm and a True Random Generator together. This is because the rate that true random numbers are generated is too slow compared to the speed that the program requires these numbers. Due to this the way the program operates is that the number generated by the TRNG are used as a seed in the PRNG and when a new one is generated it replaces the seed in the PRNG and the process is repeated as much as necessary. This means the program quickly gets a plethora of numbers with high entropy.

D. Cryptography

Cryptography in its most basic form is the conversion of a piece of text into unintelligible text and vice versa. It is most commonly used for transmitting encrypted data to and from where only those with the appropriate key or keys can decrypt and read it. The main goals of cryptography are confidentiality, authentication, non-repudiation, and availability. Confidentiality is to make sure that the data is only accessible to the parties with authorisation. Authentication in question being that the user is identifiable and not a false user unknown by the message sender or receiver. The non repudiation being that the sender or receiver are incapable of denying the message and that the computer is always able to authenticate the users. This type of security is commonly used in most modern day messaging applications.

This process is primarily to protect data from theft or alteration but is also applied in terms of a user's authorisation to a service. Cryptography is almost entirely done through True Random numbers though there are alternative options such as Cryptographic Pseudo Random Number Generator as well as using True numbers to constantly generate seeds for a PRNG to use while the TRNG generates the subsequent number that will be used as the next seed [1].

Most of the cryptographic applications are dependent on Random Number Generators for functions such as key generation, nonces and salts in signature schemes. They are extremely prevalent in most aspects of the subject due to how prevalent security is in the field and strong entropy is such a factor. An example of example of such would be an encryption algorithm that utilises Symmetric key cryptography [3]. A symmetric key is where the key in question is used for both encryption as well as decryption. The method used was to first use advanced substitution, an algorithmic approach which acts as a cipher, and it will then be further encrypted using with the use of a symmetric key, which acts as the private key, where after it will finally be passed through a symmetric key algorithm. This should prove to show the amount of computing power is used to encrypt messages and the use of entropy is key in every step of creating the encrypted message through all three of the steps.

E. Media

Media such as digital music or video services or in specific games adopt Random Numbers Generators to a great degree though in the process make some very intriguing alterations. For music services like iTunes or Spotify they have a fake random for the shuffle service. The phenomenon is called Hot Hand which was named after a study about how basketball players felt they were more likely to score a shot after scoring with multiple prior shots but it was in fact the opposite in reality where they were substantially less likely. A similar effect appeared with the shuffle feature during the introduction of these applications. The problem that arose for these music platforms was that if songs from the same artist supersede each other people didn't feel as though the random feature was fully working. The way Spotify overcame this hurdle was to take songs from the same artist and would somewhat evenly distribute these songs across the playlist when shuffled. Apple had a similar but inherently simpler in which when picking the next song in after one played it would attempt to pick a song that was by a different artist as to the one preceding it.

Music platforms are not the only medium to have altered random to fit their consumer base. In video games there is a long history of randomness adopted through card games, turn based games and rouge likes to name a few, with some very interesting techniques used throughout all of these. One of the more intriguing was a technique brought up with perceived probabilities in series such as Xcom or Fir Emblem. In these games interactions between two opposing forces are often decided by probabilities that have a percentage sign that shows the likelihood of success between from the interaction. The problem that arose was that people are very bad at understanding odds due to cognitive biases. Due to this these games will likely lie about the probability that a certain action will succeed in such a way that benefits the player. For example, if it portrayed a ninety percent chance of success in a certain scenario, it may end up being a ninety nine percent chance of success in reality, despite what the numbers might have suggested previously.

F. Overview

The implementation of Random Number Generators is to maximise the performance when in comparison to the security necessary. If security is not necessary then they will implement a simple Pseudo Random Number Generator and when security issues arise ,if possible will only use True Random number Generators to generate the seed.

IV. Entropy Techniques for random numbers

A. Pseudo Random Number Generators

1) Middle-square method: The original entropy technique was originally described in 1949. The entropy technique is flawed as in the result will after enough cycles will repeat the same pattern and due to this is not used much in the modern day. The idea behind the method is to enter a number of X digits where X is greater than 0 as the seed. The next step is to square the seed and take the middle X values. Sometimes it is necessary to add zeros at the start under the circumstances that there are fewer than 2x digits.

2) Lehmer Random Number Generator and the Linear congruential generator: The Lehmer Random Number Generator was the next step in entropy techniques through algorithmic means and was soon followed up by the Linear congruential approach that took a very similar approach. The Lehmer approach uses the formula:

$$X_{n+1} = aX_n modm$$

In which the seed, being X0 is the seed, and preferably a prime number, and a is the multiplier. This was improved upon by the Linear congruential generator but expands upon it adding an increment in c as seen:

$$X_{n+1} = (aX_n + c)modm$$

This method is the most influential Random Number Generator and is able to exceed the limits of the Lehmer algorithmic approach in which it is more free to pick it's seed as its not limited to merely prime numbers for best results.

3) Linear-feedback shift register: Linear-feedback shift register are used primarily when in conjunction with an XOR gate for its linear function and will works with bits in most instances though there are non binary applications of this method. The idea about how it functions is best demonstrated though an example as seen:

$$X_n = (X_{n-2} + X_{n-3})modm$$

where m is commonly 2. This continues until a repeating sequence appears and the program will continue to do this so no longer a need to compute. The base idea is that previous bits may be passed through an XOR gate to obtain a new bit that subsequently will be shifted left and later be used to determine a future bits value. Though this is not the best random number generator it has niche uses such as in the noise sequences such as the creation of Pseudo Random noise or for use in a scrambler that replaces specific sequences into separate sequences selectively.

4) Other Pseudo Random Entropy Techniques: Since then most future techniques in this area either have had very niche uses or further expanding on an already defined technique. Even given this, some interesting ideas have been implemented. An example of an improvement is in a Permuted congruential generation which applies a permutation to the output of a linear congruential generator to create a practical and statistically sound result.

Another example of said improvement is the implementation of Sophie German Primes to Pseudo Random Number generators. A Sophie German Prime is a number that if doubled and incremented by one produces a prime number. The application of such numbers has been seen implemented in the Monte Carlo simulation as well as uses in cryptology.

A method that is only recently being expanded upon is Counter-based random number generator in which it is based off of states. The original state is given as a seed and the output by the Counter-based random number generator will be a new state with a number. The new state will then be placed in the program and the process will repeat as many times as required.

B. Embedded Systems

Computers sometimes have built in mechanisms on order to produce entropy or have decentralised systems for a similar results.

1) Entropy from Linux Kernel: A kernel is best described as a computer program that is at the core of a computer's operating system. An early example of one is found through the Linux kernel which generates entropy by observing the timings on the keyboard and mouse movements as well as the independent device timings. The manner this is done is that 4096 bits are held as the kernel entropy pool. As the pool is diminished as the random numbers are being used they need to be replaced. This is due to when random numbers are repeating they tend to become extremely predictable as a reoccurring pattern appears. The first method is the internal clock in which whenever you interact with something the clock value is recorded with these interactions being primarily key presses or mouse clicks.

2) Windows: Microsoft's Windows during Windows 95 and all prior releases makes use of CryptoAPI and specifically CryptGetRandom for its source of entropy. This is a secure pseudorandom number generator that contains a multitude of random number generation. It gathers these random numbers from your current session. These include the tick count necessary for boot up, the current time, process and thread ids and CPU counters.

C. True Random Number Generators

For true random it is necessary to utilise a machine that generates these True random number. These are most commonly decentralised from as though True random is always preferable, due to the long generation time it is not used unless the random is truly necessary primarily due to privacy reasons.

1) Thermal Noise True Random Number Generator: Thermal noise is a very effective source of entropy. This entropy is gathered through the thermal noise of a resistor [4]. The machine itself can be split into three distinct parts. These being constructing the noise source, amplifying it and finally quantifying the waveform into a bit stream. The first step, the construction of the noise source, is calculated through the voltage between two terminals of a resistor which is random and entirely dependent on the temperature and resistance with no correlation to the current.

The next step is the amplification process where the noise source is amplified making efforts that the noise from the amplifier itself is kept at a minimum. The next and final step is quantifying the noise into a bit stream. At this point a comparator is used where there is a reference point and depending on whether the wavelength is above or below this reference decides if it will be stored as either a 0 or a 1 value. When these three tasks are completed in conjunction then there is an entropy machine that produces True Random numbers.

2) Radio Astronomy: The collection of True Random Numbers from celestial sources is an alternative means to entropy generation though they commonly are primarily used as seeds for a pseudorandom number generator [2]. The method is using either antennae or radio telescopes for the purpose of detecting the emissions of celestial sources such as star clusters or nebulae. Most astrological objects give off some form of radiation though some give off greater emissions in the likes of pulsars or quasars. These signals are truly random astronomical observations are impossible to replicate and as such are an excellent source of entropy. With the use of an amplifier these noise waves are then read and assigned bit values where they will be used as keys in a pseudo random number generator.

V. Security and Performance

The key factors when working with entropy techniques that must be always adhered to under most tasks are security, performance, speed and size. Both speed and size have become less important factors due to improvement in computing over the years. With speed becoming less a problem due to as long as it can produce randomness quicker than it is needed it is unnecessary to improve and size, specifically algorithmically, becoming less of a factor due to computer memory improving in size as well as reducing in cost. Though both size and speed are important to still acknowledge the factors that stood out as the primary areas of focus for the future are security and performance. These two are commonly at odds with each other due to improving one hinders the other and so it is about finding a balance of the two.

To get the greatest security would be to use a TRNG but the flaw with this is that the performance of the application is severely reduced due to how long it can take to generate these numbers severely slowing the program down. On the other hand, using a Linear Congruential Generator would be unadvisable for key generation or another cryptographic application due to how little security the algorithm provides. The most common solution that was applied for this is to use a TRNG to generate the seeds for a cryptographically secure pseudorandom number generator, a RNG that even when the algorithm is known it is very hard if not impossible predict the subsequent or even following output. The reason of using this method is that the performance of the program isn't hurt as new random numbers are still created while the program is still waiting for a new seed from the TRNG to generate, keeping both high performance as well as high security.

VI. Possible Future of RNG

The future of Random Number Generators is very much dependent on the applications that need it. At this time there are only two significant factors in the development of entropy and that's how random it is and the speed it is produced. There are niche algorithms out there as well as hard to obtain entropy sources which are underused as they fit a specific niche but for the most part for the future of Random number generation it will be heavily dependent on the possibility of speeding up the process while keeping high levels of entropy.

VII. Conclusion

There are many different sources of entropy with just as many applications. The application is very much tied with the task that they are presented with as well as the entropy required to complete the task in question. Sources of randomness are plenty and as time has went on finding these sources has become easier and easier with time though the application of these will most likely only be implemented outside niche uses if they improve performance without hindering security or vice versa.

References

[1] P. van Oorschot A. Menezes and S. Vanstone h. *Handbook of Applied Cryptography*. CRC Press, 1996.

[2] Tim Natusch Erin Chapman, Jerina Grewar. *A Celestial sources for random number generation*. Edith Cowan University, 2016.

[3] S. Gomathi. *A cryptography using advanced substitution technique and symmetric key generating algorithm*. IEE-EXplore, 2014.

[4] Chen Hongyi Huang Zhun. *A Truly Random Number Generator Based on Thermal Noise*. ResearchGate, 2001.

[5] Michał Burdajewicz Andrzej Rybarczyk Karol Gugała, Aleksandra Świetlicka. *Random number generation system improving simulations of stochastic models of neural cells*. SpringerLink, 2013.