Random Numbers Generation

What is a Random Number?

A random number is a number generated by a process, whose outcome is unpredictable and which cannot be subsequently reliably reproduced.

Example: The number of customers arriving in a bank daily is a random number.

The example above is considered as a true number generation (TNG), the computer must use some external physical variables that is unpredictable, such as radioactive decay of isotopes or airwaves static.

Another type of random number generation is called Pseudo Random Number Generation (PRNG), Pseudo Random Number Generation (PRNG):

Software generated random numbers only are pseudorandom. They are not truly random because the computer uses an algorithm based on a distribution and are not secure because they rely on deterministic predictable algorithms. A seed number can be set to replicate the "random" numbers generated. it is possible to predict the numbers if the seed is known. Pseudorandom number generation in everyday tools such as Python and Excel are based on the Mersenne Twister algorithm.An example use of PRNGs is in key stream generation. Stream ciphers, encrypt plaintext messages by applying an encryption algorithm with a pseudorandom cipher digit stream (key stream). Key streams of some block cipher modes, such as AES CTR (counter) mode, act as a stream cipher and can also be regarded as pseudorandom number generation.

Important properties of good random number generations:

1- Fast.

2- Portable to different programming software.

3- Have sufficiently long cycle.

4- Replicable.

There are many techniques to generate Pseudo Random Numbers. The following shall be detail:

- Mid-Square Algorithm.
- Linear Congruential Generator(LCG).

Mid-Square Algorithm:

Suppose a 4-digit seed is taken. seed = 4765 Hence, square of seed is = 4765 * 4765 = 22705225 Now, from this 8-digit number, any four digits are extracted (Say, the middle four). So, the new seed value becomes seed = 7052 Now, square of this new seed is = 7052 * 7052 = 49730704 Again, the same set of 4-digits is extracted. So, the new seed value becomes seed = 7307 . . . . This process is repeated as many times as a key is required. Mid square technique takes a certain number of digits from the square of a number. This number extracted is a pseudo-random number, which can be used as a key for hashing.

Algorithm:

1. Choose a seed value. This is an important step as for same seed value, the same sequence of random numbers is generated.

2. Take the square of the seed value and update seed by a certain number of digits that occur in the square.

Note:

The larger is the number of digits, larger will be the randomness.

3. Return the key.

```
Example:
Consider the seed to be 14 and we want a two
digit random number.

Number --> Square --> Mid-term

14     --> 0196   --> 19
19     --> 0361   --> 36
36     --> 1296   --> 29
29     --> 0841   --> 84
84     --> 7056   --> 05
05     --> 0025   --> 02
02     --> 0004   --> 00
00     --> 0000   --> 00
```

In the above example, we can notice that we get some random numbers 19,36,29,84,05,02,00 which seem to be random picks, in this way we get multiple random numbers until we encounter a self-repeating chain. We also get to know a disadvantage of this method that is if we encounter a 0 then we get a chain of 0s from that point. Also, consider that we get a random number 50 the square will be 2500 and the midterms are 50 again, and we get into this chain of 50, and sometimes we may encounter such chains more often which acts as a disadvantage and because of these disadvantages this method is not

practically used for generating random numbers.**Example:** Find the sequence of arandom numbers using (P RNG) if we considered that the seed value is 50 , 60.

1. 50^2

2500

2. 50^2

2500

1. 60^2

3600

2. 60^2

3600

This case is called Cycling and this is the on the disadvantages of the Mid-Square Method.

Linear Congruential Generator (LCG):

A linear congruential generator is a pseudorandom generator that produces a sequence of numbers x

x_1

,x_2

,x_3

,... according to the following linear recurrence:

$$x_t = a x_{t-1} + b \pmod{n}$$

Example. Let us take, for example, a=3, b=5, n=17, and *x*= 2; the sequence produced by the linear congruential generator will then be

11,4,0,5,3,14,13,10,1,8,12,7,9,15,16,…Such a generator is easy to implement, and passthe following statistical tests: frequency test, serial test, poker test, runs test, autocorrelation test, and Maurer's universal statistical test. Hence, it can beconsidered as a good candidate for generating strong pseudorandom sequences. So, let n be a large integer; choose two integers a and b; select seed x0

Sequence:

$x_i = ax_{i-1} + b \pmod{n}$

x1 = a . x0 + b (mod n)

x2 = a . x1 + b (mod n)

x3 = a . x2 + b (mod n)

.

.

Ex: Let n = 123, a =5, b = 2. Choose x_0 = 73.

$x_i = ax_{i-1} + b \pmod{n}$

x1 = 5 . (73) + 2 =365 + 2 = 367 (mod 123) = 121

x2 = 5 . (121) + 2 = 607 (mod 123) = 115

x3 = 5 . (115) + 2 = 577 (mod 123) = 85

x4 = 5 . (85) + 2 = 427 (mod 123) 58

In order to get a cipher text or sequence derived from the obtained pseudorandom numbers we can follow the following system: We have to produce a random bit sequence, so depending on the example above we obtained the following random bit sequence:

$x_0 = 73, x_1 = 121, x_2 = 115, x_3 = 85, x_4 = 58$

$x_0 = 73 \pmod 2 = 1$

$x_1 = 121 \pmod 2 = 1$

$x_2 = 115 \pmod 2 = 1$

$x_3 = 85 \pmod 2 = 1$

$x_4 = 58 \pmod 2 = 0$

.

.

the bit sequence is:

1,1,1,1,0,..........