# Practical Assignment 4

Divyansh Chaturvedi
18074005
CSE IDD

**The following two PRNGs have been used :**

**Linear Congruential Generator :**
The Linear Congruential Generator generates pseudo-random numbers in the following manner :

$$N_{i+1} = (mul*N_i + delta) \% modulus$$

Here,
$N_i$ is the i-th random number,
$N_0$ is the initial value/seed,
mul, delta and $N_0$ must be less than modulus.

Parameters Used :
$N_0$ = 56
modulus = 14641
mul = 35
delta = 617

**Park-Miller Generator :**

The Park–Miller random number generator, sometimes known as the Lehmer random number generator, is a kind of random number generator. A random number generator (RNG) of this type's general formula is $X_{k+1} = a * x_k \bmod m$. Where m is a prime number or a power of a prime number, **a** is an element of high multiplicative order modulo m, and the seed $X_0$ is coprime to m.

Parameters Used :
N = 2145678965
a = 763214
b = 88844
c = 7766

**The following randomness tests have been used :**

**Chi squared test**

Alternative and null hypothesis are as follows:
$H_0$ (null) : Data is produced randomly
$H_1$ (alternative): Data is not produced randomly

The chisquare() method from the scipy.stats package was used to implement this. We must first compute the frequency of each produced random number, and this list of frequencies is used as input by the chisquare algorithm. The output is the z-test statistic and the p-value.

**Runs test**

Null and alternative hypothesis are as follows:
$H_0$ (null) : Data is produced randomly
$H_1$ (alternative): Data is not produced randomly

The runstest_1samp() function from the statsmodels package was used to implement this. The input is a list of generated random numbers, and the output is the z-test statistic and p-value.
When the p-value is larger than or equal to 0.05 (threshold value), we must accept the null hypothesis, and hence numbers are produced at random.

**Code :**
The code takes input the number of random numbers from the user that are to be generated in a variable named total. Then it uses the Linear Congruential Generator(LCG) to produce random numbers. Then it run both runs test and chisquare test over it . After this, it uses the Park-Miller Generator to produce random numbers. Then it run both runs test and chisquare test over it .

```
from statsmodels.sandbox.stats.runs import runstest_1samp
from scipy.stats import chisquare
```

```python
print("Enter the number of random numbers needed: ")
total =int(input())
mod_lcg = 14641
lcg_seed = 56
mul = 35
deltaal_val = 617
num=lcg_seed
lcg_list = []
for j in range(total):
  new_num = (mul*num + deltaal_val)%mod_lcg
  lcg_list.append(new_num)
  num = new_num
#runs test
print("Result of runs test for LCG: ", runstest_1samp(lcg_list)[1])
#chisqauretest
freq_lcg = {}
for i in range(mod_lcg):
  freq_lcg[i]=0
lcgF_list = []
for i in lcg_list:
  freq_lcg[i]=freq_lcg[i]+1
for key,value in freq_lcg.items():
  lcgF_list.append(value)
print("Result of chisquare test for LCG: ",chisquare(lcgF_list)[1])
n = 2145678965
a = 763214
b = 88844
c = 7766
seed = 12345678

def uniform(seed):
    hi = seed // b
    lo = seed - (b * hi)
    t = (a * lo) - (c * hi)
    if t > 0:
        seed = t
    else:
        seed = t + n
    return seed

A = []
cnt = total
for i in range(0,cnt):
    seed = uniform(seed)
    A.append(seed)
```

```
#runs test
print("Result of runs test for park miller : ", runstest_1samp(A)[1])
#chisqauretest
freq_pmt = {}
for i in range(mod_lcg):
  freq_pmt[i]=0
pmF_list = []
for i in lcg_list:
  freq_pmt[i]=freq_pmt[i]+1
for key,value in freq_pmt.items():
  pmF_list.append(value)
print("Result of chisquare test for park miller : ",chisquare(lcgF_list)[1])
```

Output :

```
Result of runs test for LCG:  0.10118559757604385
Result of chisquare test for LCG:  0.9999999991998727
```

```
Result of runs test for park miller :  0.22657166630260817
Result of chisquare test for park miller :  0.9999999991998727
```

**Results**:

On Generating 1000 random numbers For LCG , the Runs test gives a result of 0.101 which is greater than 0.05 and Chisquare test gives a result of 0.99>0.05; so we fail to reject the null hypothesis and hence numbers are generated in a random fashion. For the Park-Miller test , the Runs test gives a result of 0.22>0.05 and Chisquare test gives a p-value of 0.99>0.05; so we fail to reject the null hypothesis and hence numbers are generated in a random fashion.

Github Link : [Network Security PA 4]