Assignment 3 Report

Running the code:

To run the code, use the following command

**python3 assignment3.py <number of output bytes> <Number of runs> <name of plot>**

This code will run the code as given in the assignment 3 and save the plot generated during the compilation. E.g.

**python3 assignment3 2 20 plot\_2\_20.png**

Plots and Observation:

Chart, histogram

Description automatically generated

Fig 1: Plot for randomness using 2bytes outputs with 60 runs.

Chart, histogram

Description automatically generated

Fig 2: Plot for randomness using 4bytes outputs with 60 runs.

Chart, histogram

Description automatically generated

Fig 3: Plot for randomness using 8bytes outputs with 60 runs.

Chart

Description automatically generated

Fig 4: Plot for randomness using 32bytes outputs with 60 runs.

Chart, histogram

Description automatically generated

Fig 5: Plot for randomness using 128bytes outputs with 60 runs.

Chart, histogram

Description automatically generated

Fig 6: Plot for randomness using 1024bytes outputs with 60 runs.

As we can see from the above plots the randomness on outputs ranging from short through long (i.e., 2 bytes, 4 bytes, 8 bytes, 32 bytes, 128 bytes, 1024 bytes) is increasing as the value of R is decreasing. This is due to the fact that in the pseudoRandomGeneration() function we are shuffling the bits with more number of time for 1024bytes as compared to 2bytes.

Another observation we saw that is if we decrease the number of runs then the plot we are getting is very jumbled and if we increase the number of runs then the plot we are getting is quiet intuitive and values are not changing a lot.

Q : How many bits need to be flipped (on average) before the differential bitstream looks random?

A : For fig1, fig 2, and fig 3 we can conclude that we are getting good amount of randomness after flipping at least 3 random bits. For fig4, fig5 and fig6 we can conclude that we are getting good amount of randomness after flipping at least 2 random bits. We assume this trend is correct as for fig4, fig5 and fig6 we are shuffling the outputs by 32, 128 and 1024 times respectively which means we will achieve randomness quiet early and vice versa for fig1, firg2, and fig3.