CS6500: Assignment 1

Cryptanalysis of RC4 encryption algorithm

Sanjana Prabhu EE17B072

March 4, 2021

1 Introduction

In this assignment, we perform a cryptanalysis of the RC4 encrpytion algorithm. The claim is that upon toggling a few bits in the key, half of the output bits may not change as expected. This is what is tested by means of a randomness test on the output of RC4 in this assignment.

2 Implementation

This RC4 algorithm was first implemented in Python and for each pair of keys with a difference in one or more bits, two outputs were collected from the RC4 algorithm and XORed together. This represents the difference of the two streams. Now, we run randomness tests on this XORed stream to check the randomness of the RC4 algorithm and the variation of randomness with increasing number of bits toggled and increasing output length. The figures obtained are shown in the next section.

3 Plots of randomness vs number of bits toggled

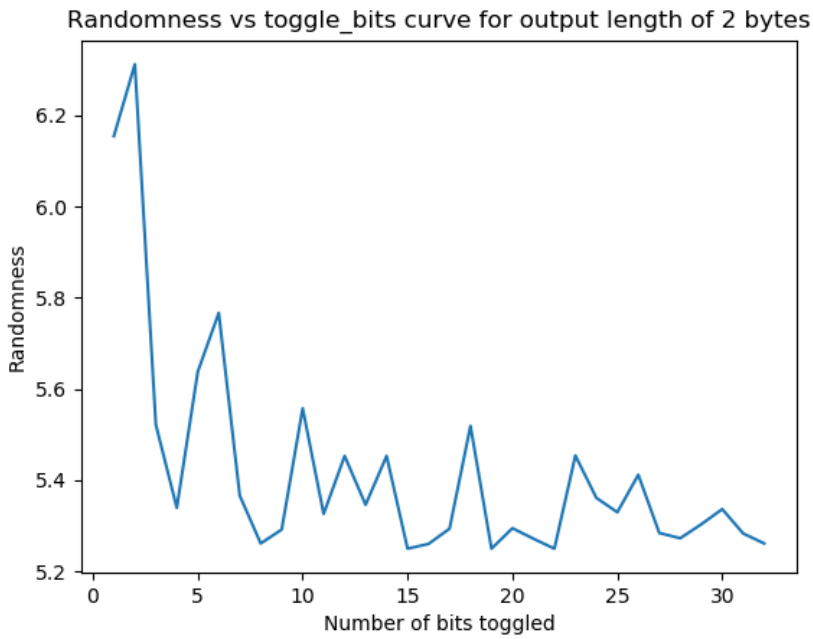

Figure 1:

4 Conclusion

- As shown by the figures, the randomness of the outputs increases with increasing number of bits toggled as well as with increasing length of the output stream. Note that lower values of randomness(Y axis values in the figures) implies that the two bit streams are more random.
- As the length of the output bit stream increases, the effect of toggling one or more bits in the key would propagate and result in the two output streams being more random.
- As the number of bits toggled in the key increases, it is expected that the internal state of the RC4 is more different for the two keys and this difference is demonstrated in the lower randomness values for higher number of bits toggled.
- If the RC4 algorithm were perfectly random, one would expect the two output streams generated by the two keys to be fairly independent of each other. That is, their difference(the XOR output of the two streams) would be similar to an iid sequence of 0s and 1s. This is true for higher values of output lengths/bits toggled. One such instance is demonstrated in Figure 7. For output length = 1024, by toggling 16 bits in the key, we run the randomness test and capture the counter

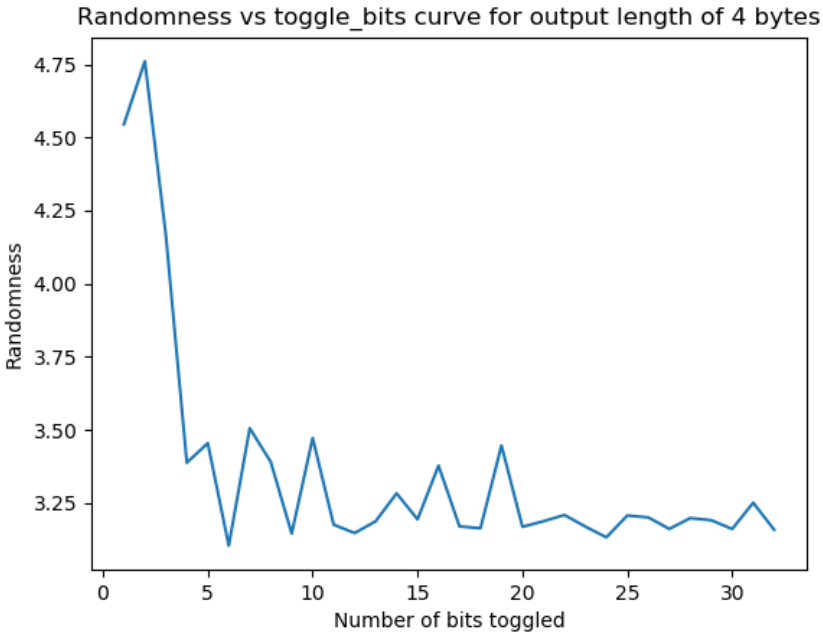

Figure 2:

arrays at every iteration. If the difference stream were similar to an iid sequence of 0s and 1s, one would expect that all the 256 numbers of the counter would be generated by the bitstream, uniformly. This is seen in Figure 7.

5 What I learnt from this assignment

I learnt that the RC4 algorithm is not perfectly random and two similar keys generate slightly similar outputs. From a network security point of view, this is an inherent weakness as it would be quite easy for the attackers to perform cryptanalysis and determine the key and hence the message.

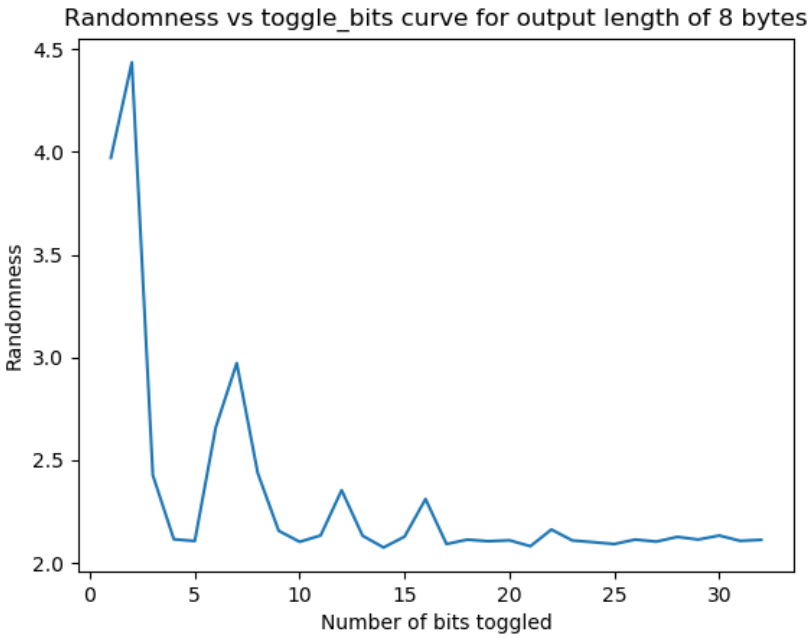

Figure 3:

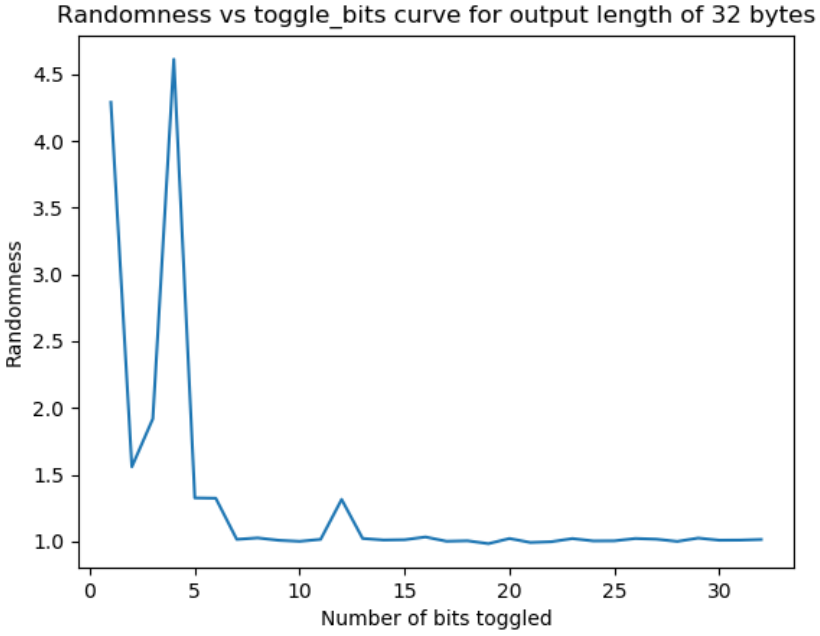

Figure 4:

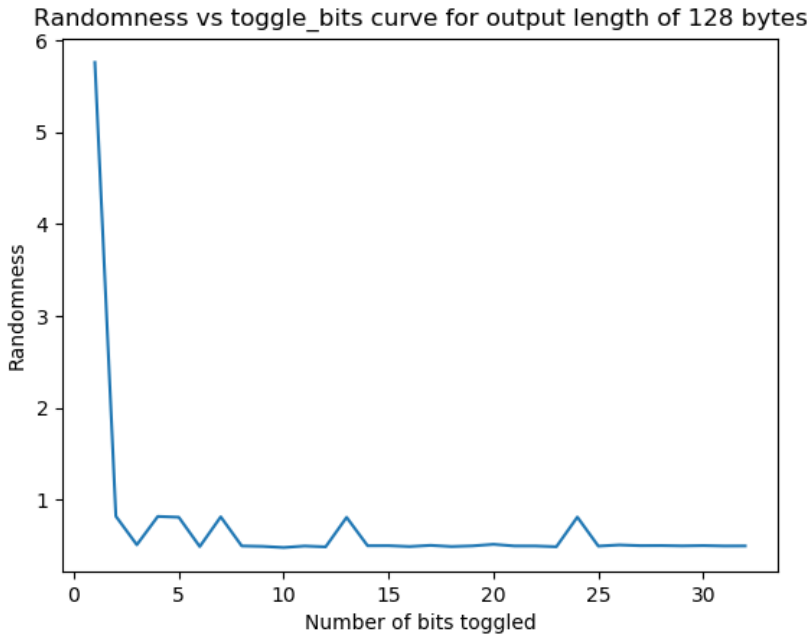

Figure 5:

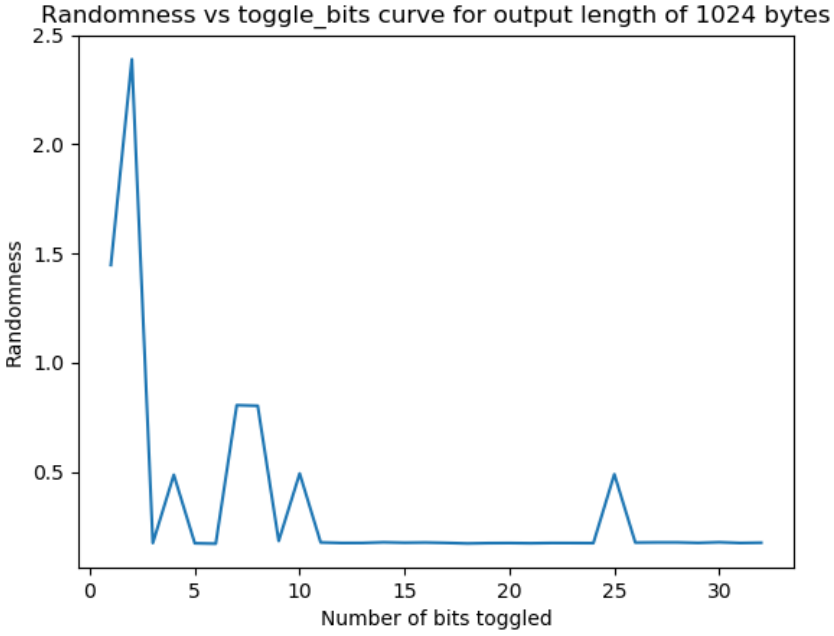

Figure 6:

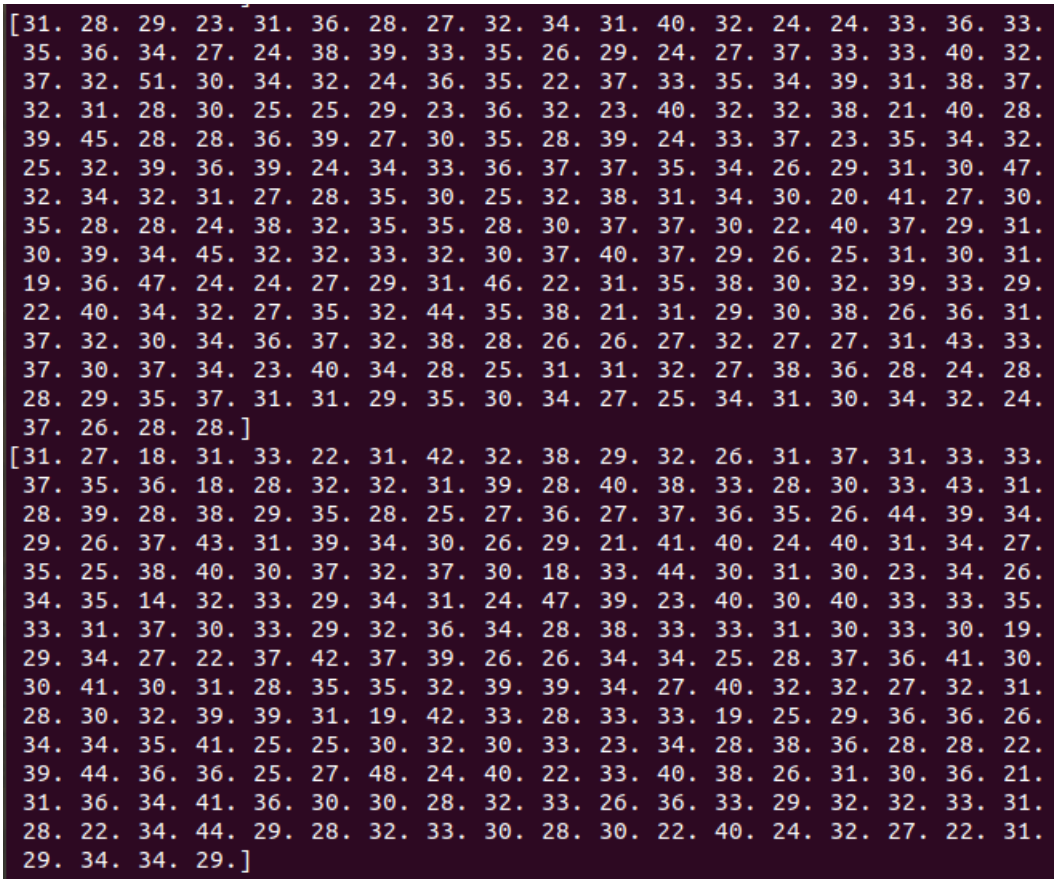

Figure 7: Counter values for output length 1024 and number of toggle bits = 16