Technical Report
CS6500 Network Security : Programming assignment 1
Name : Kolan Nikshith Reddy
Roll No : CS20m030

Cryptanalysis of Stream Cipher RC4:

RC4 (Rivest Cipher 4) is a Stream Cipher, while it is remarkable for its simplicity and speed in software, multiple vulnerabilities have been discovered in RC4, rendering it insecure. It is especially vulnerable when the beginning of the output keystream is not discarded, or when non- random or related keys are used.

After analysing the differential output bits for randomness using the given formula using a simple frequency counting test and computing a numerical measure of the randomness like so:

• N = number of samples. • C = number of counters. • D = standard deviation of counter values.
• R =(D ∗ C)/N;

The closer the randomness (R) is to zero, the more random the data.

In my implementation, I checked the RC4 cryptanalysis for 4 cases with the help of 2 parameters.

Parameter 1: Whether the bits toggled are from 1 to 2048 position or the toggled bits are in the last portion of the key.

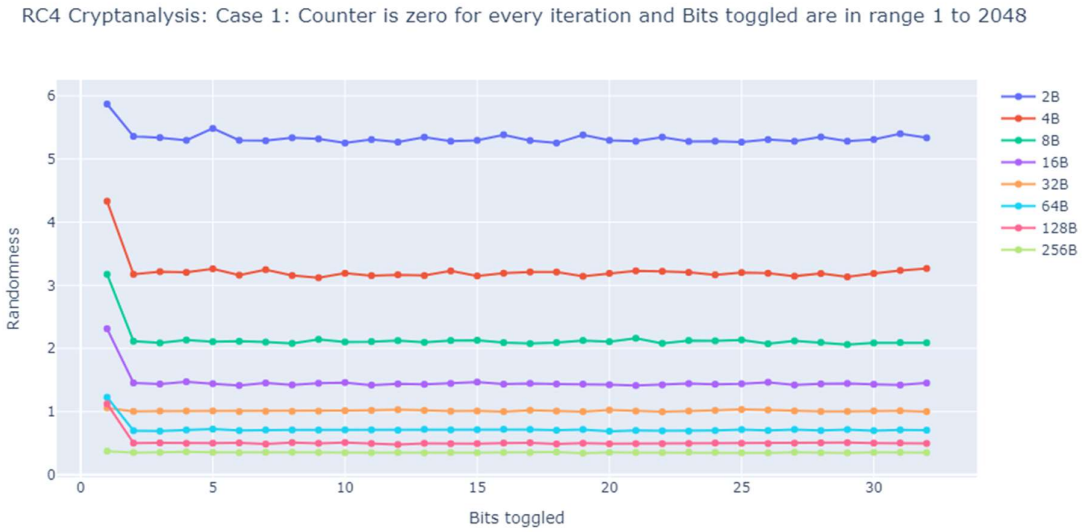
Parameter 2: Using two types of counters. One changes with every iteration and one Changes only once for each bit toggled.

1. Counter is initialised to zero once for every iteration of 40 iterations and bits toggled can be in range 1 to 2048
2. Counter is initialised to zero once for every iteration of 40 iterations and bits toggled can be in range 1500 to 2048
3. Counter is initialised to zero once for each bit toggled and bits toggled can be in range 1 to 2048
4. Counter is initialised to zero once for each bit toggled and bits toggled can be in range 1500 to 2048

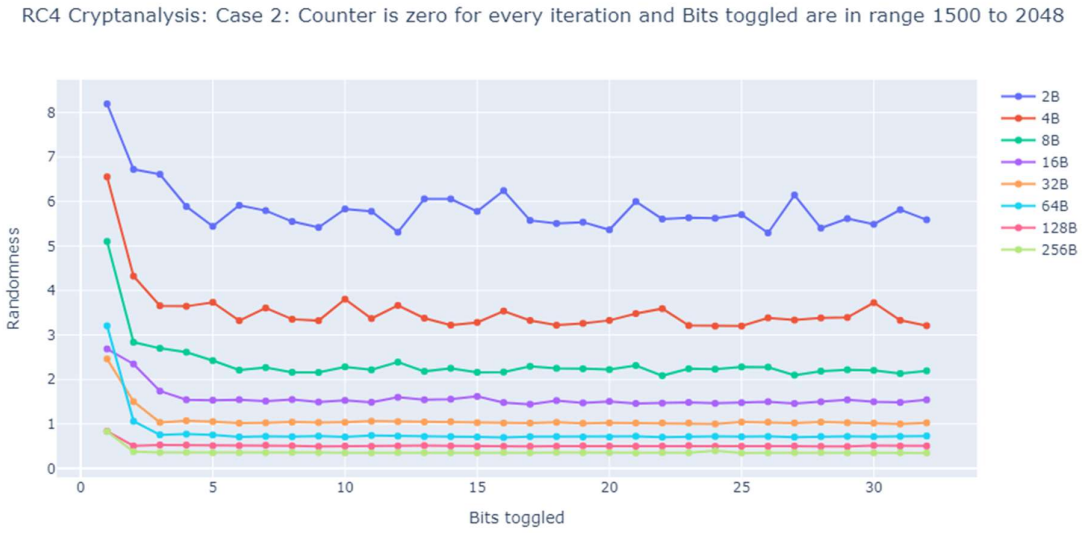
Observations:

1. By Measuring the randomness on outputs ranging from short through long (i.e from 2 B to 256 B) in the above experiment, I found out that,
    a. Randomness is getting close to zero with moving from 2B to 256B. That is randomness of data is increasing with the increase in size.

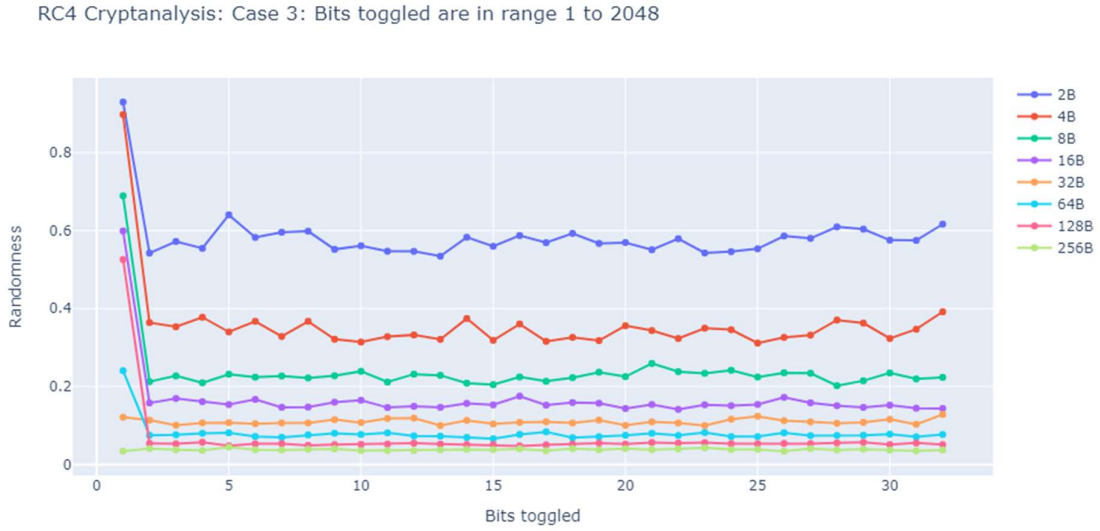
Case 1: When bits are toggled from the first bit itself, we got a neat curve of randomness. i.e, Randomness is increasing with increase in no of bits toggled.



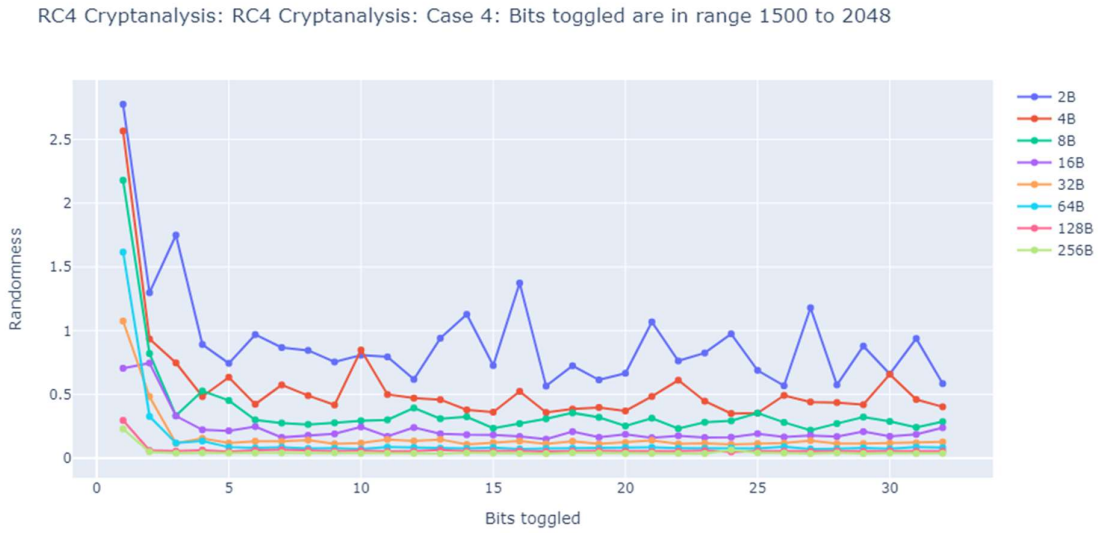
case 2: When the bits toggled are in the last portion of the key, The randomness is not neat. That is we have similarities in the cipher text



Case 3: Even with Different counter, When bits are toggled from the first bit itself, we got a neat curve of randomness. i.e, Randomness is increasing with increase in no of bits toggled

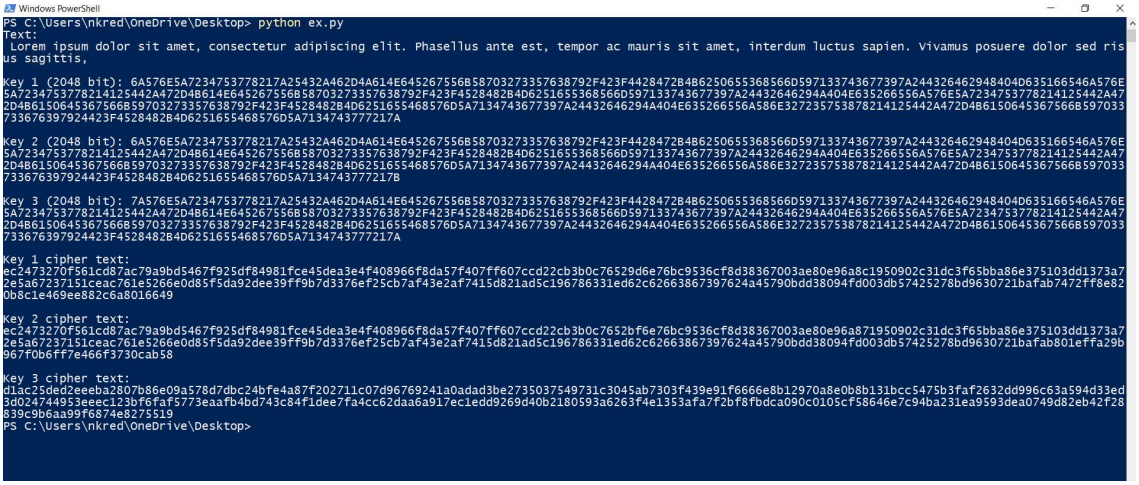


Case 4: Even with Different counter, When the bits toggled are in the last portion of the key, The randomness is not neat. That is we have similarities in the cipher text.



Observation 2:
Key 1 : 2048 bit key is considered
Key 2 : only last one bit of key is altered from key 1
Key 3 : only first one bit is altered from key 1

1. When 1 bit is changed in the end of the key, we can see cipher text is almost similar and is not random.
2. When 1 bit is changed in the start of the key, we can see cipher text is completely different and is random.



Observation 3:
When we are using a long key, it is better to generate some initial dummy keystream which we can drop. Here we are also checking using a long stream , so we can perform the above as we can generate some initial dummy keystream and drop it.
With this, We can improve randomness. That is, from 2B to 256B we improved randomness.

Learnings:

1. the most important weakness of RC4 comes from the insufficient key schedule; the first bytes of output reveal information about the key. This can be corrected by simply discarding some initial portion of the output stream.
2. RC4 lacks authentication and if not used together with a strong message authentication code (MAC), then encryption is vulnerable to a bit-flipping attack
3. RC4 stream ciphers are simple to use, fast, easy to implement.
4. RC4 stream ciphers are implemented on large streams of data.

From this experiment I got a good exposure of how various ciphers work and Got hands on experience with implementing a cipher and in analysing the results using various graphs.
I tried toggling random bits and found out with trial and error that that bits that need to be toggled are the ones in the starting portion of the key, in order to generate a complete random cipher text.