COL 759: Assignment 3:

Due date: 5th November 2021, midnight. Late submission with penalty is not applicable for this assignment.

The objective of this assignment is to do some simple cryptanalysis of the RC4 encryption algorithm. In this assignment you will make some differential measurements of randomness for related keys, and plot graph of your results.

Details

First implement the RC4 algorithm (you can also download the code). With a stream cipher like RC4, it should be the case that, if you flip exactly one bit in the key, 50% of output bits should flip. If you look closely at the key setup, it doesn't look like this is the case. It appears that single-bit differences in the key will result in fairly small differences in the internal state of RC4. After enough output, this small change will eventually propagate and then you would expect 50% of the output bits to flip.

You have to study the random bits. Make sure you can accept keys of length up to 2048 bits.

Generate the output bits from two runs and XOR them

Generate a random key of 2048 bits. Collect some output from RC4. Flip one or more of the key bits. Reinitialize RC4 and collect the output again. XOR them together. This gives you the difference of the two streams. If RC4 were perfect, this difference stream would be perfectly random.

Analyse the differential output bits for randomness

Implement a simple frequency counting test for randomness testing. Create an array of counters of length equal to a power of two (say, 256). Now, say the test data is a sequence of bits b0b1b2b3...bN. You can look at each sequence of 8 bits (b0 … b7, b1 … b8, b2 … b9) as a number and increment the appropriate counter in the array. When you're done you would expect that the counters would be approximately equal. If the two original bitstreams were very similar, you would expect the counters for lots of zeros to have higher values than the counters for lots of ones.

You can compute a numerical measure of the randomness like so:

N = number of samples

C = number of counters

D = standard deviation of counter values

$R = (D \times C) / N$

The closer the randomness (R) is to zero, the more random the data. You might consider using different numbers of counters and see if your randomness measure changes very much.

Run this randomness test lots of times and collect the results

Measure the randomness on outputs ranging from short through long (i.e., 2 bytes, 4 bytes, 8 bytes, 32 bytes, 128 bytes, 1024 bytes). For each of these, you need to consider the effect of

flipping one bit in the key, two bits in the key, and so forth through 32 bits. For each of these pairs, you should make at least 20 measurements of the randomness and average the results.

Graph the results

You should produce a graph with one line for each output length. The Y axis is the randomness (higher values imply less randomness). The X axis is the number of bits you flipped. You would expect each line to start somewhere above zero and, as you flip more bits, approach zero quickly. You would also expect to see higher values for the lines corresponding to shorter runs of data.

Discuss

How many bits need to be flipped (on average) before the differential bitstream looks random? This tells you something about the maximum useful key length for RC4. If you use a 128-bit RC4 key, each key bit is replicated 16 times in the internal key. If flipping 16 bits isn't enough to generate a random differential output, then that would imply 128-bit keys are not usefully strong.

If a vendor wished to ship a product using RC4 to encrypt short messages (50 bytes max), perhaps they should throw out some of the initial output from RC4 to let the key bits mix properly. How much?

Another analysis

Rather than randomness tests, you might observe that two RC4 systems initialized with similar keys may initially generate identical values but will eventually diverge. Try to measure the average length (in bytes) of identical output as a function of the number of bits you change in the key.

While you can measure that value, you may also be able to derive it probabilistically. See what you can do.