Lab 3 – The RC4 Stream Cipher

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key- size stream cipher with byte-oriented operations which is based on the use of random permutations.

Review the pdf file posted under module 7 in canvas to assist you in working on this lab.

Download and inspect the python file **RC4\_1.py**

The Python code in this file includes the following

* Initialization of the S vector
* Generating the T vector
* A key generation algorithm that includes the necessary initial and subsequent permutations on S which in turn provides the keystream necessary to encrypt (and decrypt) a certain message
* An EXOR encryption algorithm that utilizes the generated keystream
* An EXOR decryption algorithm

There 2^n Blocks of n bits in regular

In this system, there are 2^3 blocks of 3 bits

1. While the RC4 cipher is in essence a byte-oriented stream cipher, the code given in the downloaded program assumes a 3-bit oriented cipher for simplicity. What are the values (show all in hexadecimal) of the initial S vector?
2. The Python code also provides the following plaintext and key:

Plaintext: 001010010010111010011000

Key: 1010010000011101

What is the value of the T vector that is generated from this key?

1. What is the keystream that is generated from the permutations?
2. What is the ciphertext that results from the EXOR operation?
3. Adjust the code such that it assumes:
4. a 4-bit oriented cipher with the following plaintext and key

Plaintext: AABBCCDD

Key: 456

S Vector:

T Vector:

Keystream:

1. a 5-bit oriented cipher with the following plaintext and key

Plaintext: AABBCCDDEEFF

Key: 456789

S Vector:

T Vector:

Keystream:

Repeat steps 1-4 above with these values.

1. Adjust the code to function as a true RC4 byte-oriented stream cipher. Using the same plaintext and key as in 5b above, what is T? What is the ciphertext? Is this system formidable against brute force attacks? If not suggest a key that would make it so.

I would say that this system does have a flaw in that there is a pattern that can be deciphered via cryptanalysis. Because the key is exclusively integers, and sequenced integers at that, cryptanalysis, specifically in regards of the T, is fairly trivial and brute force attacks even more so. Thus, I would recommend a key of random ASCII characters so that the amount of characters needed to be tried for a brute force attack makes the system computationally secure.