RC4

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. The algorithm is based on the use of a random permutation. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. RC4 is used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers.

- A variable length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], …., S[255].
- S contains a permutation of all 8-bit numbers from 0 through 255.

Initialization of S

S[0] = 0, S[1] = 1, …………., S[255] = 255 .

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

If the length of the key K is 256 bytes, then K is transferred to T. Otherwise, for a key of length *keylen* bytes, the first *keylen* elements of T are copied from K, and then K is repeated as many times as necessary to fill out T.

Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:

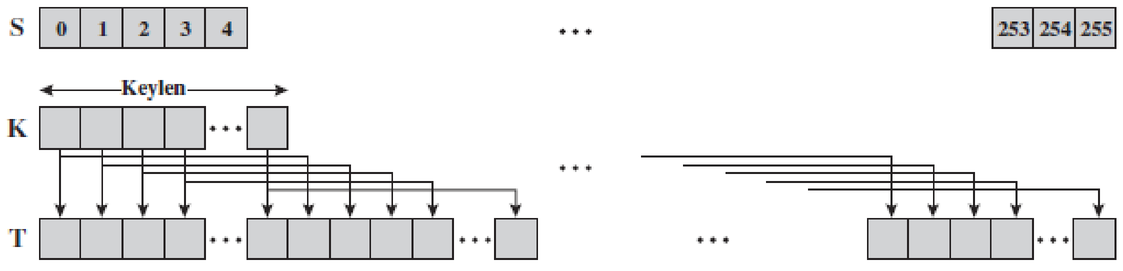
```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
        j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
```

Stream Generation: Pseudo-random generation algorithm (PRGA) Scrambling

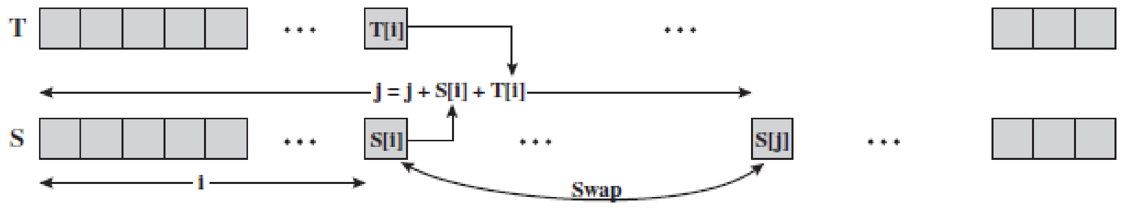
The output byte is selected by looking up the values of S[i] and S[j] adding them in modulo 256, sum is used as index. S[S[i]+S[j]%256] is used as byte of the key stream.

- Increments i
- Looks up ith element of S, S[i] and adds to j
- Exchange S[i] and S[j], then uses S[S[i]+S[j]%256] for key stream byte
- Bitwise XOR with next byte of message to generate cipher byte.

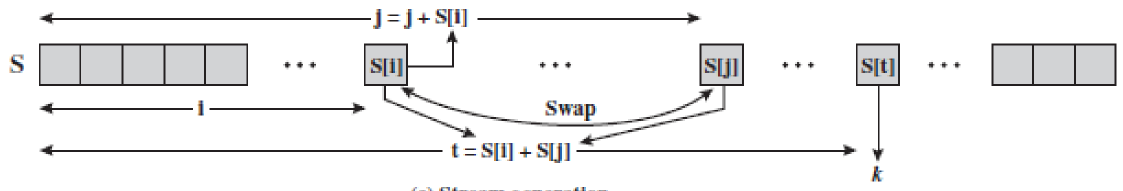
```
/* Stream Generation */
i, j = 0;
while (true)
        i = (i + 1) mod 256;
        j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream generation