RC-4

- Stream Cipher.
- Extremely simple!
- Very fast – especially in software
- Easily adapts to any key length (1 byte to 256 bytes)
- Used in SSL / TLS
- WEP
- (Was) protected by trade secret – exposed (anonymously posted on the web) in 1994
Key Initialization
K[0] ....K[keylen-1] --- key bytes
For i = 0 to 255
  – S[i] = i;
  – T[i] = K[i mod keylen];
j = 0;
For i = 0 to 255
  – j = (j + S[i] + T[i]) mod 256;
  – SWAP(S[i], S[j]);
Throw away T, K; (retain S)
RC-4 – Stream Generation

- \(i, j = 0;\)
- while (true)
  - \(i = (i+1) \mod 256;\)
  - \(j = (j + S[i]) \mod 256;\)
  - SWAP(S[i], S[j]);
  - \(t = (S[i] + S[j]) \mod 256;\)
  - \(k = S[t];\)
- The vector \(S\), at any time, is a random permutation of 1 to 256 (only swap performed on the vector).
void code(long* v, long* k) {
  unsigned long y=v[0], z=v[1], sum=0, /* set up */
  delta=0x9e3779b9, n=32;               /* a key schedule constant */
  while (n-->0) {                       /* basic cycle start */
    sum += delta;
    y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
    z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;   /* end cycle */
  }
  v[0]=y ; v[1]=z ; }

void decode(long* v, long* k) {
  unsigned long n=32, sum, y=v[0], z=v[1],
  delta=0x9e3779b9 ;
  sum=delta<<5 ;
  /* start cycle */
  while (n-->0) {
    z-= (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
    y-= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
    sum-=delta ;  }
  /* end cycle */
  v[0]=y ; v[1]=z ;  }

Input v – 64 bits
As two 32 bit quantities
v[0], v[1]

k – 128 bits
As four 32 bit quantities
k[0],k[1],k[2],k[3]