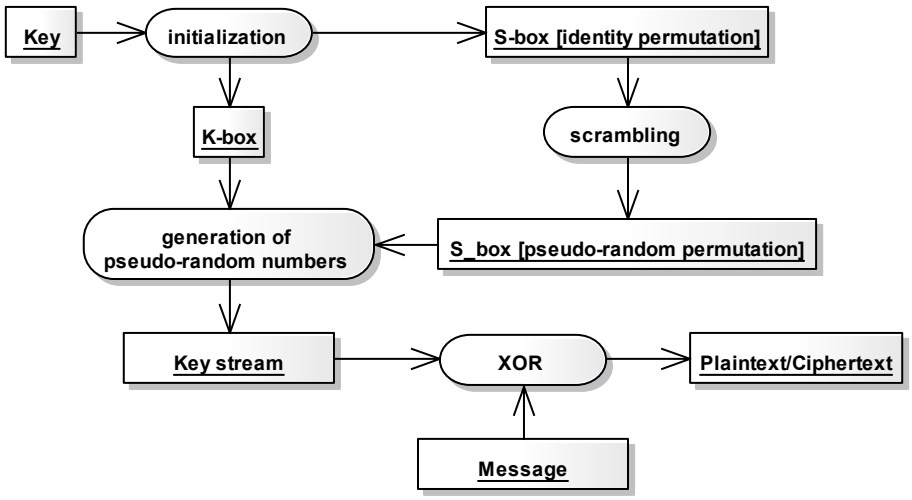
Implement a symmetric encryption/decryption algorithm according to the specification below.



1. Initialization

There are two 256-byte arrays, the state and the key array (denoted "S-box" and "K-box" respectively). The S-box is filled linearly, such as S[0] = 0, S[1] = 1, ..., S[255] = 255. Later, it is used to generate a new pseudorandom number, for each input byte that we want to encrypt/decrypt. K-box is filled with the key, repeating the key until the array fills up (the key length must be between 1 and 256 characters).

2. Scrambling

The elements in the S-box are rearranged to produce a new unknown key-dependent permutation. The process is described by the following pseudo-code:

```
j := 0;
For i from 0 to 255
    j := (j + S[i] + K[i]) mod 256;
    swap(S[i], S[j]);
End;
```

3. Generation of pseudo random numbers

A pseudorandom sequence of bytes called the **key stream** is generated in a loop. For as many iterations as the size of the message, the algorithm shuffles two elements in the S-box and outputs one byte of the key stream according to the following pseudo-code:

```
i := 0;
j := 0;
For k from 0 to (MSG.length-1)
      i := (i + 1) mod 256;
      j := (j + S[i]) mod 256;
      swap(S[i], S[j]);
      output( S[ ( S[i] + S[j] ) mod 256 ] );
End;
```

4. Encryption/decryption

The key stream is combined with the message using a XOR operation to produce either ciphertext or plaintext.

```
Encryption:  Plaintext XOR KeyStream = Ciphertext
Decryption:  Ciphertext XOR KeyStream = Plaintext
```