Blockchain

Nepal

Symmetric Key Ciphers

XOR

- XOR

  0+0 = 0;

  0+1 = 1;

  1+0 = 1;

  1+1 = 0

- XOR of bit vectors (words)

  >> we do XOR for each corresponding bit pairs

  e.g., 0011 + 1010 = 1001

One Time Pad (OTP)

- Key: sequence of random bytes: k1k2...kl

- Encryption:
  - message is represented as sequence of bytes : m1m2...ml
  - cipher text = message XOR key
    m1m2...ml + k1k2..kl = c1c2...cl
    where ci = mi+ki for all i=1,...,l

- Decryption:
  - ciphertext represented as sequence of bytes : c1c2...cl
  - plaintext = ciphertext XOR key
    c1c2...cl+k1k2...kl = m1m2...ml
  - Proof: ci+ki = mi+ki+ki = mi

Properties of OTP

- Perfect secrecy
  >> Looking at the encrypted messages provides no information about the original message.

- Large key size
  >> Need a truly random key that has the same length as the message

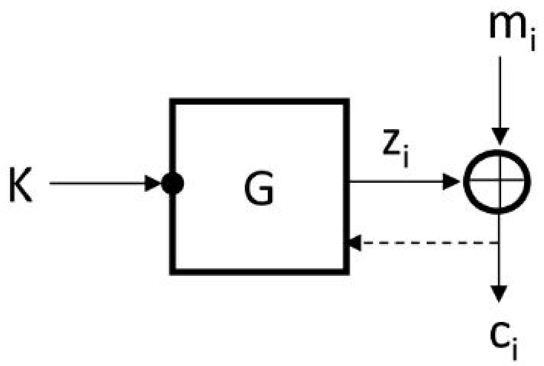
- impractical in many applications
  >> how to send the key in a secure way to the recipient?
  >> key management is a pain

Stream Ciphers

Idea: simulate the truly random key stream of the one-time pad with a pseudo random sequence generated from a random seed.

Terminology:

- mi - plaintext character
- ci - ciphertext character
- zi - key-stream character
- K - key(seed)
- G - key stream generator



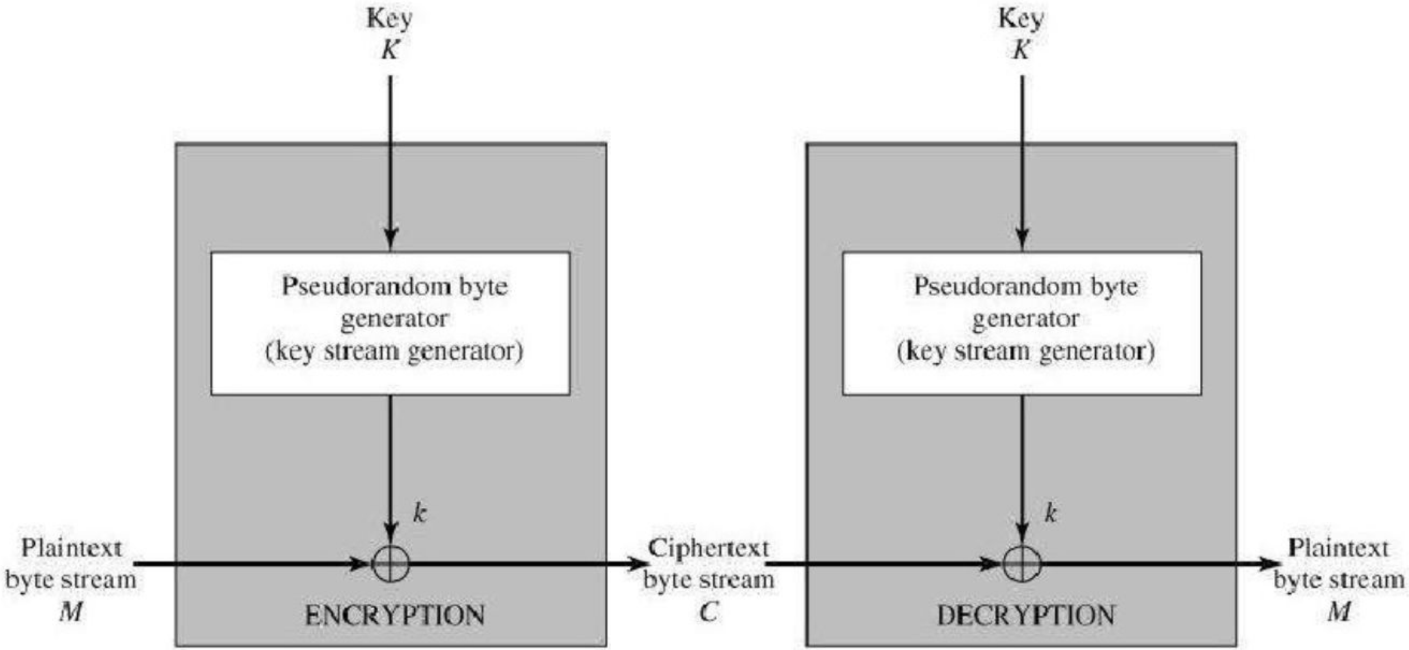
Examples:- LSFR, RC4

Properties of Stream Ciphers

- Usually very efficient
  >> Fast (hardware impl)
  >> Requires less memory

- Ciphertext always has the same length as the plaintext

RC4


Key
K
Pseudorandom byte
generator
(key stream generator)
k
Plaintext
byte stream
M
ENCRYPTION
Ciphertext
byte stream
C
Key
K
Pseudorandom byte
generator
(key stream generator)
k
DECRYPTION
Plaintext
byte stream
M

RC4

1. Initialize an array of 256 bytes

2. Run the Key Scheduling Algorithm (KSA) on them

3. Run the Pseudo-random Generation Algorithm (PRGA) on the (KSA) output to generate Key stream

4. XOR the data with a key stream

RC4- Algorithm

- **initialization:**

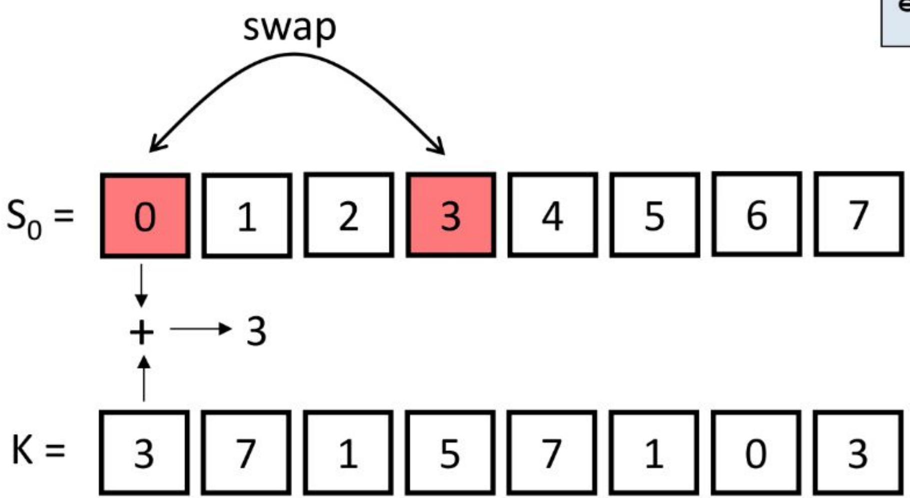
```
for i = 0 to 255 do
  S[i] = i
end

j = 0
for i = 0 to 255 do
  j = j+S[i]+K[i mod len(K)] mod 256
  swap(S, i, j)
end

i = 0
j = 0
```

- **generation:**

```
i = i+1 mod 256
j = j+S[i] mod 256
swap(S, i, j)
return S[ S[i]+S[j] mod 256 ]
```

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```

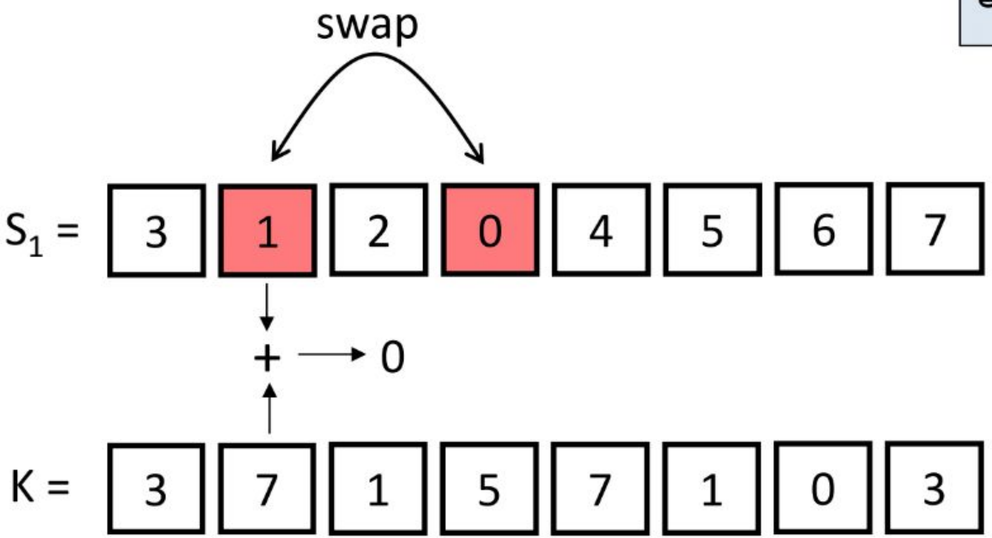


$j = 0$

$i = 0$ $\quad j_0 = 3$

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```



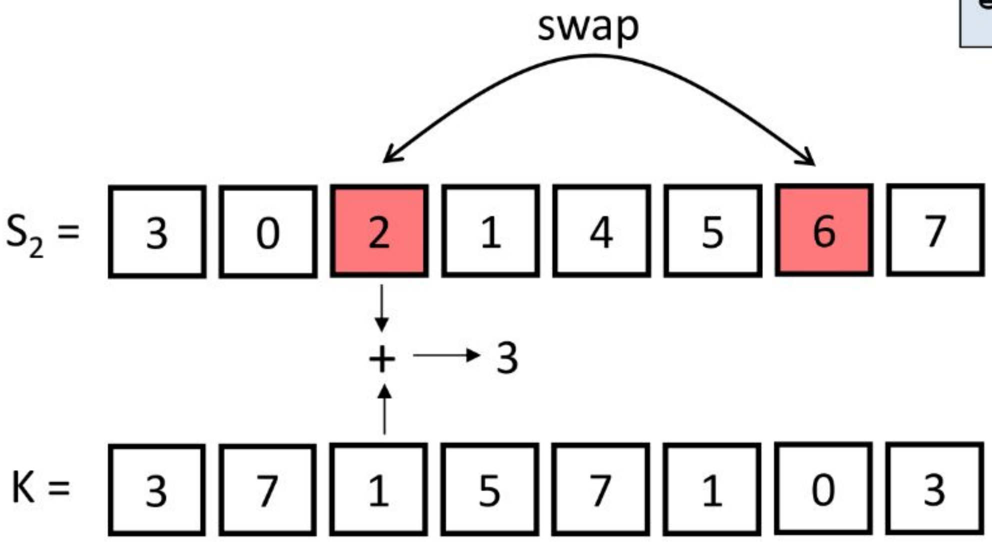
$j = 0$

$i = 0$ $\quad j_0 = 3$

$i = 1$ $\quad j_1 = 3$

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```



$j = 0$

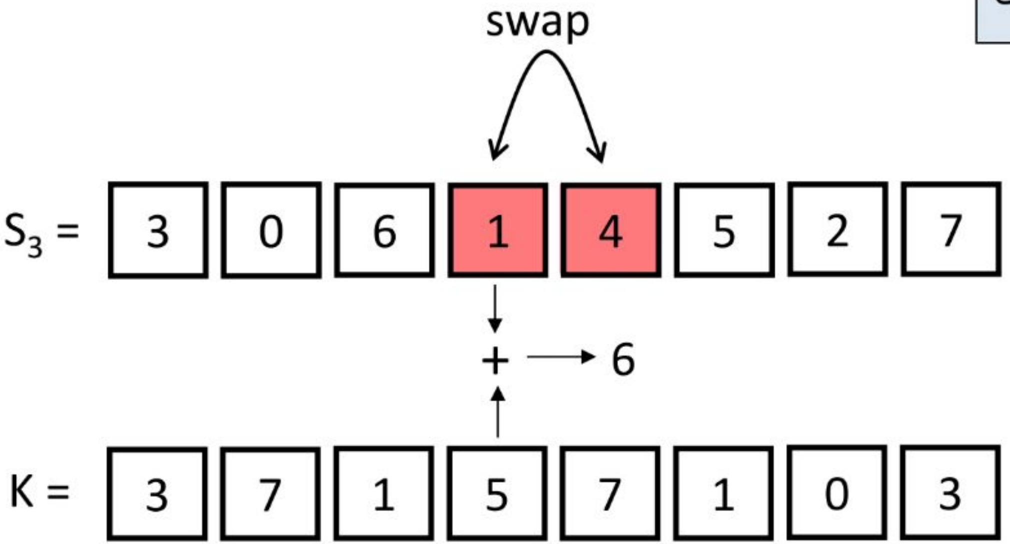
$i = 0 \quad j_0 = 3$

$i = 1 \quad j_1 = 3$

$i = 2 \quad j_2 = 6$

Example: RC4 Initialization

```
for i = 0 to 7 do
     j = j + S[i] + K[i] mod 8
     swap (S, i, j)
end
```

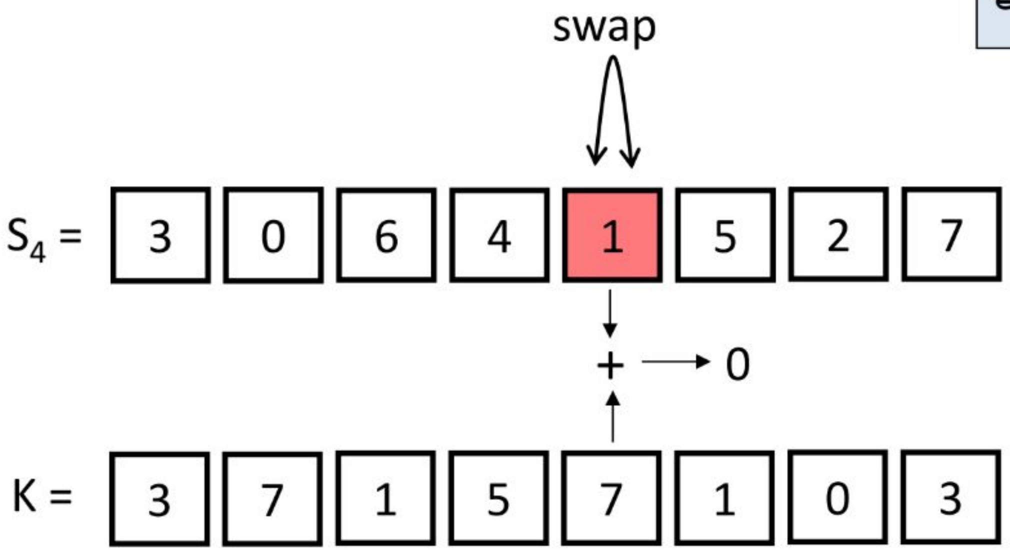


$j = 0$

$i = 0$	$j_0 = 3$
$i = 1$	$j_1 = 3$
$i = 2$	$j_2 = 6$
$i = 3$	$j_3 = 4$

Example: RC4 Initialization

```
for i = 0 to 7 do
     j = j + S[i] + K[i] mod 8
     swap (S, i, j)
end
```

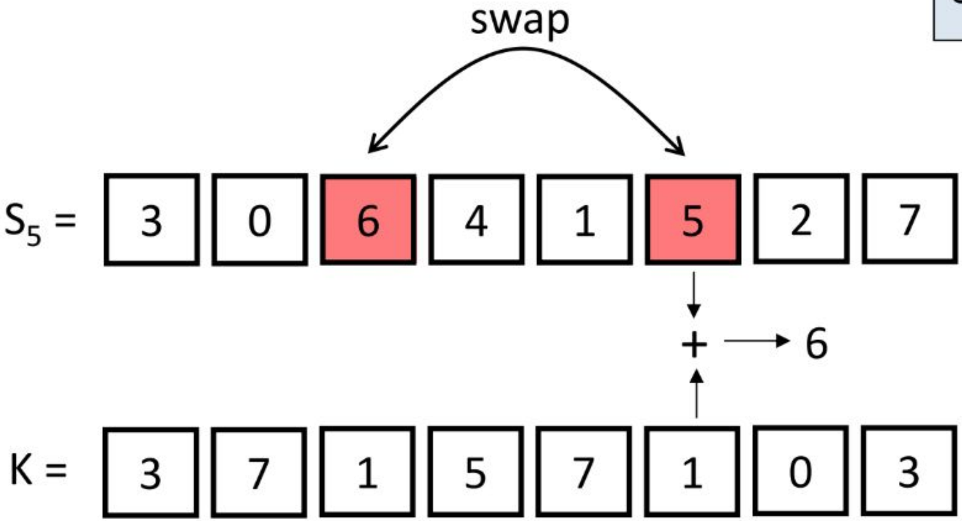


$j = 0$

$i = 0$	$j_0 = 3$
$i = 1$	$j_1 = 3$
$i = 2$	$j_2 = 6$
$i = 3$	$j_3 = 4$
$i = 4$	$j_4 = 4$

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```

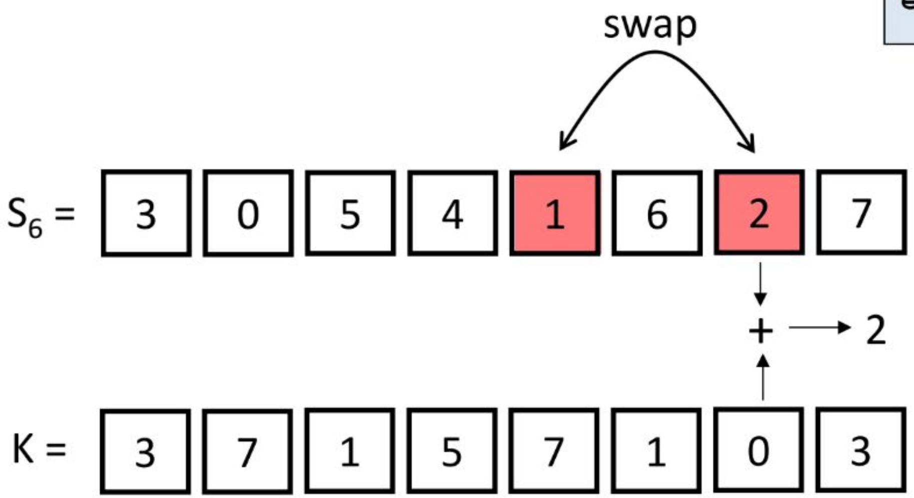


$j = 0$

$i = 0$	$j_0 = 3$
$i = 1$	$j_1 = 3$
$i = 2$	$j_2 = 6$
$i = 3$	$j_3 = 4$
$i = 4$	$j_4 = 4$
$i = 5$	$j_5 = 2$

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```

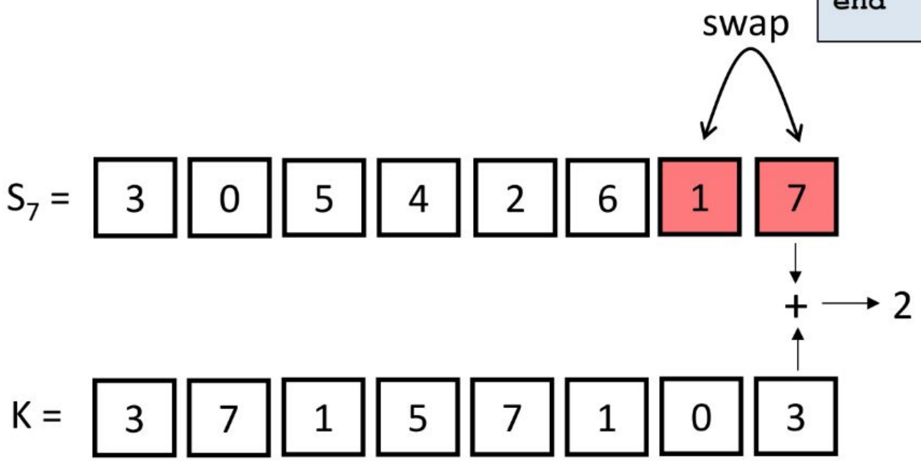


$j = 0$

$i = 0$	$j_0 = 3$
$i = 1$	$j_1 = 3$
$i = 2$	$j_2 = 6$
$i = 3$	$j_3 = 4$
$i = 4$	$j_4 = 4$
$i = 5$	$j_5 = 2$
$i = 6$	$j_6 = 4$

Example: RC4 Initialization

```
for i = 0 to 7 do
    j = j + S[i] + K[i] mod 8
    swap (S, i, j)
end
```

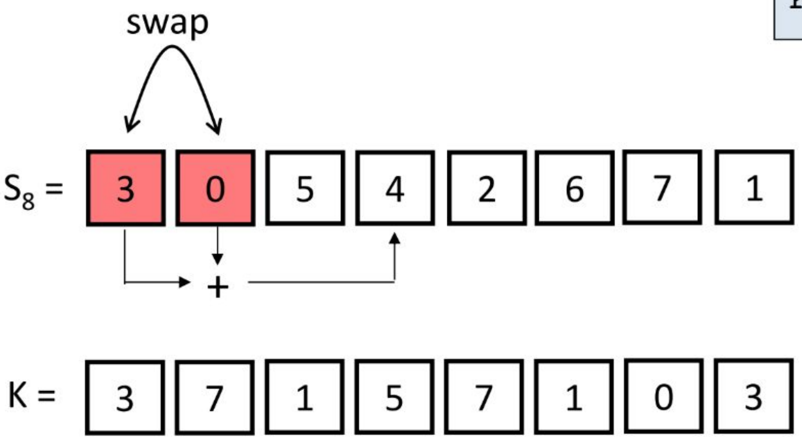


$j = 0$

$i = 0$	$j_0 = 3$
$i = 1$	$j_1 = 3$
$i = 2$	$j_2 = 6$
$i = 3$	$j_3 = 4$
$i = 4$	$j_4 = 4$
$i = 5$	$j_5 = 2$
$i = 6$	$j_6 = 4$
$i = 7$	$j_7 = 6$

Example: RC4 Generation

```
i = i+1 mod 8
j = j + S[i] mod 8
swap (S, i, j)
return S[ S[i]+S[j] mod 8 ]
```



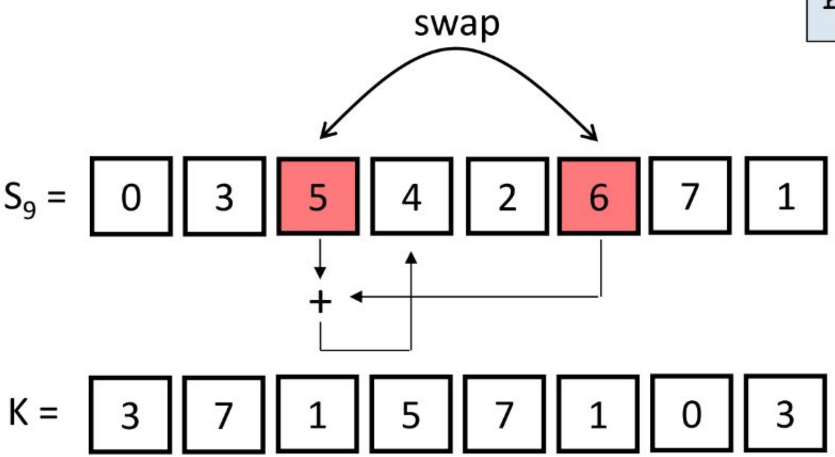
$i = 0 \qquad j = 0$

$i = 1 \qquad j_1 = 0 + S[1] = 0$

$output_1 = S[3] = 4$

Example: RC4 Generation

```
i = i+1 mod 8
j = j + S[i] mod 8
swap (S, i, j)
return S[ S[i]+S[j] mod 8 ]
```



$i = 0 \qquad j = 0$

$i = 1 \qquad j_1 = 0 + S[1] = 0$

$i = 2 \qquad j_2 = 0 + S[2] = 5$

$output_2 = S[5 + 6 \bmod 8] = S[3] = 4$

Exercise 2

> Implement RC4 Stream cipher and implement proper unit tests for encryption and decryption.