Encryption Algorithms & Protocols

Symmetric key Crypto

- Stream Ciphers

Dr. Omar Abusada
E-mail: abossada1@gmail.com

Symmetric key Crypto

- Stream cipher is based on **one-time pad cipher**.
  - Except that key is relatively short
  - Key is stretched into a long keystream
  - Keystream is used just like a one-time pad.
- Block cipher is based on **codebook concept**
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employs both “confusion” and “diffusion”

Stream cipher

- Once upon a time, not so very long ago, stream ciphers were the king of crypto
- Today, not as popular as block ciphers
- We'll discuss two stream ciphers…
- A5/1
  - Based on shift registers
  - Used in GSM mobile phone system
- RC4
  - Based on a changing lookup table.
  - Used in many places.

A5/1: Shift Registers Cipher

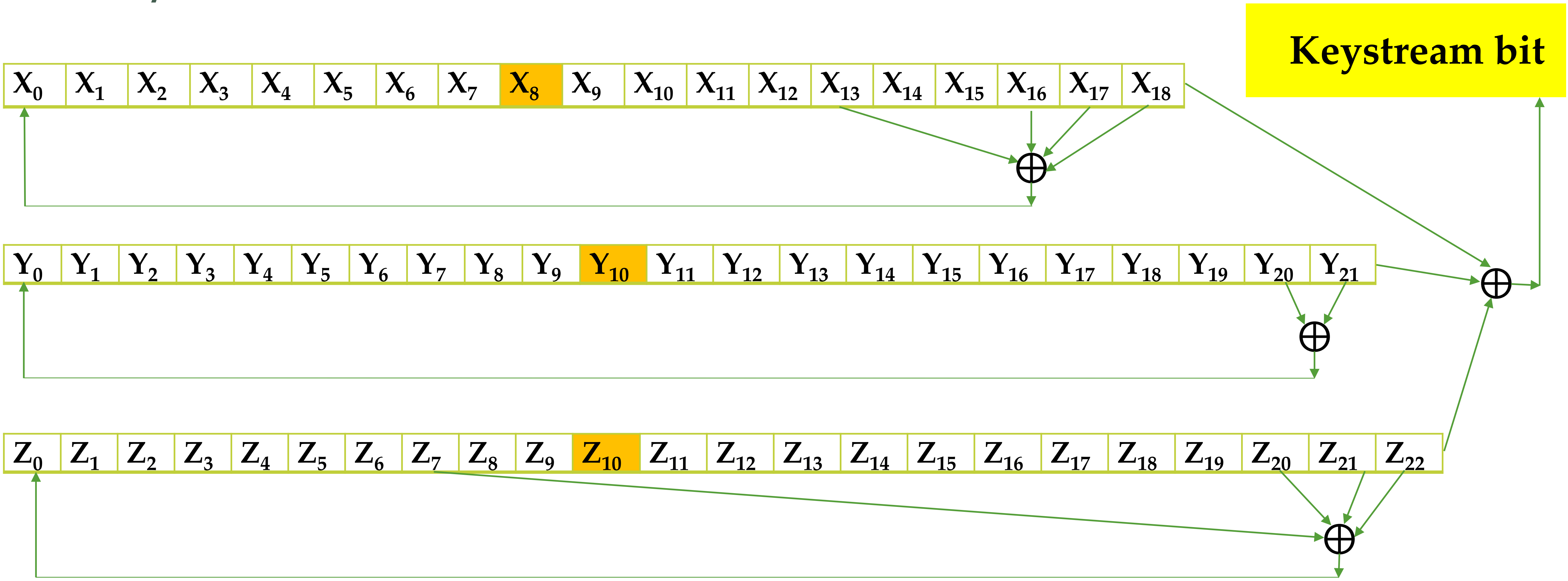
- A5/1 uses 3 shift registers
  - X: 19 bits $(x_0, x_1, x_2, \ldots\ldots, x_{18})$
  - Y: 22 bits $(y_0, y_1, y_2, \ldots\ldots, y_{21})$
  - Z: 23 bits $(z_0, z_1, z_2, \ldots\ldots, z_{22})$
  - Total: 64 bits $\longrightarrow$ key

A5/1 Rules

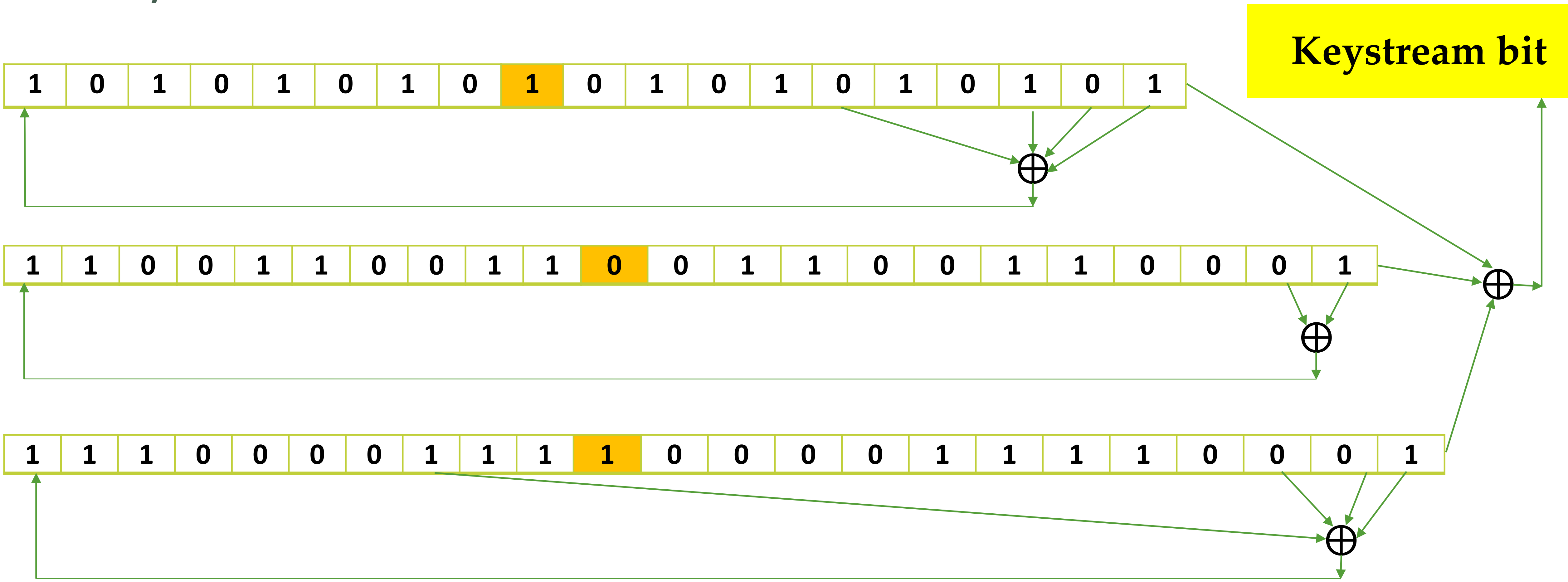
- At each step: $m = maj(x_8, y_{10}, z_{10})$
- Examples: $maj(0,1,0) = 0\ and maj(1,1,0) = 1$
- If $x_8 = m$ then X steps
  - $x_i = x_{i-1}\ for\ i = 18,17,16,15, \ldots \ldots, 1\ and\ x_0 = t$
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
- If $y_{10} = m$ then y steps
  - $y_i = y_{i-1}\ for\ i = 21,20,19,18, \ldots \ldots, 1\ and\ y_0 = t$
  - $t = y_{20} \oplus y_{21}$
- If $z_{10} = m$ then z steps
  - $z_i = z_{i-1}\ for i = 22,21,20,19, \ldots \ldots, 1\ and\ z_0 = t$
  - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$

Keystream bit is $x_{18} \oplus y_{21} \oplus z_{22}$

A5/1 Rules


Keystream bit
X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9 X_{10} X_{11} X_{12} X_{13} X_{14} X_{15} X_{16} X_{17} X_{18}
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8 Y_9 Y_{10} Y_{11} Y_{12} Y_{13} Y_{14} Y_{15} Y_{16} Y_{17} Y_{18} Y_{19} Y_{20} Y_{21}
Z_0 Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7 Z_8 Z_9 Z_{10} Z_{11} Z_{12} Z_{13} Z_{14} Z_{15} Z_{16} Z_{17} Z_{18} Z_{19} Z_{20} Z_{21} Z_{22}

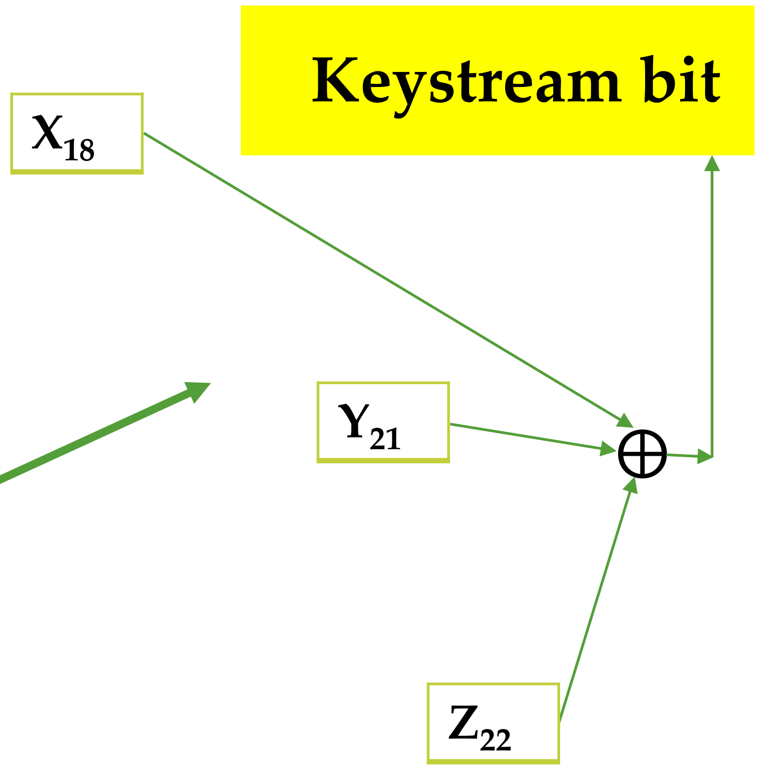
A5/1 Rules



in this example, $m = maj(x_8, y_{10}, z_{10}) = maj(1,0,1) = 1$. Register **X steps**, Y does not step, **and Z steps**

Understanding the rules

- Each variable here is a single bit.
- Key is used as initial fill of registers.
- Each register steps (or not) based on $maj(x_8, y_{10}, z_{10})$.
- Keystream bit is XOR of rightmost bits of registers.



استخدم طريقة A5/1 Keystream وذلك لتوليد مفتاح متماثل لتشفير البيانات التالية 11001. افرض ان مسجلات كل من X,Y,Z كما موضحة بالشكل أدناه. استخدم المفتاح الذي سيتم توليده وذلك لتشفير البيانات الموضحة أعلاه.

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}
0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1	0	1	1
1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1	0	1
1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1	0
0	1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1
0	1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1
0	0	1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	0

Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}	Y_{17}	Y_{18}	Y_{19}	Y_{20}	Y_{21}
0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1	1	1
0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1	1	1
0	0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1	1
0	0	0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1	1
0	0	0	0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	1
1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0

Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{13}	Z_{14}	Z_{15}	Z_{16}	Z_{17}	Z_{18}	Z_{19}	Z_{20}	Z_{21}	Z_{22}
1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	1	0	1	1
1	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	1	0	1
1	1	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	1	0
0	1	1	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0	1
1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0
1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	0	0	1	1	1	0	0	0

XOR

1- $M_1=Maj(1,0,1)=1$

X and Z move (Y no move)

X: $X_{13} \oplus X_{16} \oplus X_{17} \oplus X_{18}$

X: $1 \oplus 0 \oplus 1 \oplus 1=1$

Z: $Z_7 \oplus Z_{20} \oplus Z_{21} \oplus Z_{22}$

Z: $1 \oplus 0 \oplus 1 \oplus 1=1$

KEY1=$1 \oplus 1 \oplus 1=1$

2- $M_2=Maj(0,0,0)=0$

X,Y and Z move (All move)

X: $X_{13} \oplus X_{16} \oplus X_{17} \oplus X_{18}$

X: $1 \oplus 1 \oplus 0 \oplus 1=1$

Y: $Y_{20} \oplus Y_{21}$

Y: $1 \oplus 1=0$

Z: $Z_7 \oplus Z_{20} \oplus Z_{21} \oplus Z_{22}$

Z: $1 \oplus 1 \oplus 0 \oplus 1=1$

KEY1=$0 \oplus 1 \oplus 0=1$

3- $M_3=Maj(1,1,1)=1$

X,Y and Z move (All move)

X: $1 \oplus 0 \oplus 1 \oplus 0=0$

Y: $1 \oplus 1=0$

Z: $1 \oplus 0 \oplus 1 \oplus 0=0$

KEY1=$1 \oplus 1 \oplus 1=1$

4- $M_3=Maj(0,1,1)=1$

Y and Z move (X no move)

Y: $1 \oplus 1=0$

Z: $0 \oplus 0 \oplus 0 \oplus 1=1$

KEY1=$1 \oplus 1 \oplus 0=0$

5- $M_3=Maj(0,0,1)=0$

X and Ymove (Z no move)

X: $0 \oplus 1 \oplus 0 \oplus 1=0$

Y: $0 \oplus 1=1$

KEY1=$0 \oplus 0 \oplus 0=0$

Key=00111

Example

- Use A5/1 keystream generation method to generate a key which will be used to encrypt the data (11001). Assume that X, Y and Z registers are as shown below. Once the key is generated, use it to encrypt the data shown above.

  - More about A5/1 Please see videos below:

  - https://www.youtube.com/results?search_query=RC4+keystream+generation

  - https://www.youtube.com/watch?v=1GoP_HfF_v4

RC4 Cipher

- RC4 is a stream cipher, but it's a completely different from A5/1.
- The RC4 is optimized for software implementation, whereas A5/1 is designed for hardware.
- RC4 produces a keystream byte at each step, whereas A5/1 only produces a single keystream bit.
- Generating a byte at each step is much better than generating a single bit.
- RC4 is a self-modifying lookup table.
- Table always contains a permutation of the byte values 0,1,…,255
- At each step, RC4 does the following
  - Swaps elements in current lookup table
  - Selects a keystream byte from table

RC4 Algorithm

- *for i = 0 to 255*
- *S[t] = i*
- *K[i] = key [z mod N]*
- *next i*
- *J = 0*
- *for i = 0 to 255*
- *j = (j + s[i] + K[i]) mod 256*
- *swap(s[i],s[j])*
- *next i*
- *i = j = 0*

For each keystream byte, swap elements in table and select byte

i = (i + 1) mod 256

j = (j + S[i]) mod 256

swap(S[i], S[j])

t = (S[i] + S[j]) mod 256

keystreamByte = S[t]

Use keystream bytes like a one-time pad

Example on RC4 Cipher

- S-array (state Array) $\boldsymbol{s-array} = [\mathbf{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7}]$
- K-array (key Array) $\mathbf{k} - \boldsymbol{array} = [\mathbf{1\ 2\ 3\ 6}]$
- Plaintext (plaintext array) $\mathbf{Plaintext} - \boldsymbol{array} = [\mathbf{1\ 2\ 2\ 2}]$
- Initialize T-array with Key. $\mathbf{T} - \boldsymbol{array} = [\mathbf{1\ 2\ 3\ 6\ 1\ 2\ 3\ 6}]$ **(Same size with S-array).**
- $i = 0\ to 7$ **(8 iterations)**
- $j = (j + s[i] + k[i])\ mod\ 8$
- $swap(s[i]\ , s[j])$
- Number of iteration is depends on the size of $\boldsymbol{S\ -\ array}$

RC4 Cipher

$s = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$ $K = [1\ 2\ 3\ 6\ 1\ 2\ 3\ 6]$

- **STEP 1: Key scheduling process: (Number of iteration depends on the size of $S - array$) = (8)**

- $i = 0\ ,\ j = 0$
- $j = (j + s[i] + k[i]) mod\ 8 = (0 + 0 + 1)\ mod\ 8\ = 1$
- $j = 1\ ,\ swap(s[0]\ ,s[1]) = swap\ (0{,}1)$
- $s = [1\ 0\ 2\ 3\ 4\ 5\ 6\ 7]$
- --
- $i = 1, j=1$
- $j = (j + s[i] + k[i]) mod\ 8 = (1 + 0 + 2)\ mod\ 8 = 3$
- $j = 3\ ,swap(s[1]\ ,s[3]) = swap\ (0{,}3)$
- $s - array = [1\ 3\ 2\ 0\ 4\ 5\ 6\ 7]$
- --
- $i = 2, j=3$
- $j = (j + s[i] + k[i]) mod\ 8 = (3 + 2 + 3)\ mod\ 8 = 0$
- $j = 0\ ,swap(s[2]\ ,s[0]) = swap\ (2{,}1)$
- $s - array = [2\ 3\ 1\ 0\ 4\ 5\ 6\ 7]$

- $i = 3, j=0$
- $j = (j + s[3] + k[3]) mod\ 8 = (0 + 0 + 6)\ mod\ 8 = 6$
- $j = 6\ ,swap(s[3]\ ,s[6]) = swap\ (0{,}6)$
- $s - array = [2\ 3\ 1\ 6\ 4\ 5\ 0\ 7]$
- --
- $i = 4, j=6$
- $j = (j + s[i] + k[i]) mod\ 8 = (6 + 4 + 1)\ mod\ 8 = 3$
- $j = 3\ ,\ swap(s[4]\ ,s[3]) = swap\ (4{,}6)$
- $s - array = [2\ 3\ 1\ 4\ 6\ 5\ 0\ 7]$
- --
- $i = 5, j=3$
- $j = (j + s[i] + k[i]) mod\ 8 = (3 + 5 + 2)\ mod\ 8\ = 2$
- $j = 2\ ,\ swap(s[5]\ ,s[2]) = swap\ (5{,}1)$
- $s = [2\ 3\ 5\ 4\ 6\ 1\ 0\ 7]$

RC4 Cipher

$K = [1\ 2\ 3\ 6\ 1\ 2\ 3\ 6]$

$s = [2\ 3\ 5\ 4\ 6\ 1\ 0\ 7]$

$\boldsymbol{Initial\ state\ array\ s - array = [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]}$

$\boldsymbol{New\ state\ array\ s - array = [2\ 3\ 7\ 4\ 6\ 0\ 1\ 5]}$

- $\boldsymbol{i = 6}$, ***j=2***
- $j = (j + s[i] + k[i]) mod\ 8 = (2 + 0 + 3)\ mod\ 8 = 5$
- $j = 5\ , swap(s[6]\ , s[5]) = swap\ (0,1)$
- $s - array = [2\ 3\ 5\ 4\ 6\ 0\ 1\ 7]$
- ---
- $\boldsymbol{i = 7}$, ***j=5***
- $j = (j + s[i] + k[i]) mod\ 8 = (5 + 7 + 6)\ mod\ 8 = 2$
- $j = 2\ ,\ swap(s[i]\ , s[j]) = swap\ (7,5)$
- $s - array = [2\ 3\ 7\ 4\ 6\ 0\ 1\ 5]$

RC4 Cipher

New state array $s - array = [2\ 3\ 7\ 4\ 6\ 0\ 1\ 5]$

- **STEP 2: Stream Generation: (Number of iteration depends on the size of the Key = (4)**

- $i = (i + 1) mod\ 8$
- $j = (j + s[i]) mod\ 8$
- $swap(s[i], s[j])$
- $t = (s[i] + s[j]) mod\ 8$
- *Keystrembyte=S[t]*
- ---
- ***i=0, j=0***
- $i = (i + 1) mod\ 8 = (0+1)\ mod\ 8 = 1$
- $j = (j + s[i]) mod\ 8 = (0+3) mod\ 8 = 3$
- $swap(s[i], s[j]) = swap(s[1], s[3]) = swap(3,4)$
- $s - array = [2\ 4\ 7\ 3\ 6\ 0\ 1\ 5]$
- $t = (s[i] + s[j]) mod\ 8 = (s[1] + s[3]) mod\ 8 = (4 + 3) mod\ 8 = 7$
- $\mathbf{k[0] = S[7] = 5}$

- ***i=1, j=3***
- $i = (i + 1) mod\ 8 = (1+1)\ mod\ 8 = 2$
- $j = (j + s[i]) mod\ 8 = (3+7) mod\ 8 = 2$
- $swap(s[i], s[j]) = swap(s[2], s[2]) = swap(7,7)$
- $s - array = [2\ 4\ 7\ 3\ 6\ 0\ 1\ 5]$
- $t = (s[i] + s[j]) mod\ 8 = (s[2] + s[2]) mod\ 8 = (7 + 7) mod\ 8 = 6$
- $\mathbf{k[1] = S[6] = 1}$
- ---
- ***i=2, j=2***
- $i = (i + 1) mod\ 8 = (2+1)\ mod\ 8 = 3$
- $j = (j + s[i]) mod\ 8 = (2+3) mod\ 8 = 5$
- $swap(s[i], s[j]) = swap(s[3], s[5]) = swap(3,0)$
- $s - array = [2\ 4\ 7\ 0\ 6\ 3\ 1\ 5]$
- $t = (s[i] + s[j]) mod\ 8 = (s[3] + s[5]) mod\ 8 = (0 + 3) mod\ 8 = 3$
- $\mathbf{k[2] = S[3] = 0}$

RC4 Cipher

- **STEP 2: Stream Generation: (Number of iteration depends on the size of the Key = (4)**

$s - array = [2\ 4\ 7\ 0\ 6\ 3\ 1\ 5]$

- ***i=3, j=5***
- $i = (i+1) mod\ 8$ =(3+1) mod 8 = 4
- $j = (j + s[i]) mod$ 8=(5+6)mod 8 = 3
- $swap(s[i], s[j]) = swap(s[4], s[3]) = swap(6,0)$
- $s - array = [2\ 4\ 7\ 6\ 0\ 3\ 1\ 5]$
- t= $(s[i] + s[j])mod\ 8 = (s[4] + s[3])mod\ 8 = (0 + 6)mod\ 8$=6
- $\boldsymbol{k[3]}$***=S[6]=1***

New key $- \boldsymbol{array} =$ [5 1 0 1]

RC4 Cipher

- **STEP 3: Encryption and Decryption**

- $\textbf{New key} - \boldsymbol{array} = [5\ 1\ 0\ 1]$
- $\textbf{Plaintext} - \boldsymbol{array} = [1\ 2\ 2\ 2]$
- Now, convert both into binary then **XOR** them
- $\textbf{Plaintext} - \boldsymbol{array}\ \textbf{in binary} = [1\ 2\ 2\ 2] = [001\ \ 010\ \ 010\ \ 010]$
- $\textbf{New key} - \boldsymbol{array}\ \textbf{in binary} = [5\ 1\ 0\ 1] = [101\ \ 001\ \ 000\ \ 001]$
- **Ciphertext =Plaintext ⊕ New Key**
- **Ciphertext=**

$$\begin{matrix}[001 & 010 & 010 & 010] \\ [101 & 001 & 000 & 001]\end{matrix} \oplus$$

$$[100\ \ 011\ \ 010\ \ 011]$$

$$\boldsymbol{Ciphertext} = \mathbf{4323}$$

RC4 Cipher

- RC4 is used in many applications, including SSL and WEP.
- RC4 is sure to be a major player in the crypto arena for many years to come.
- Stream ciphers were once king of the hill, now relatively rare, in comparison to block ciphers.
- Some have even gone so far as to declare the death of stream ciphers.
- However, today there are an increasing number of significant applications where dedicated stream ciphers are more appropriate than block ciphers.
- Examples of such applications include wireless devices

RC4 Cipher Videos

- https://www.youtube.com/watch?v=lRyzKIvxNdM
- https://www.youtube.com/watch?v=7b0p-rsizGo

... Thank you ...