ICT Course: Information Security

Nguyen Minh Huong

ICT Department, USTH

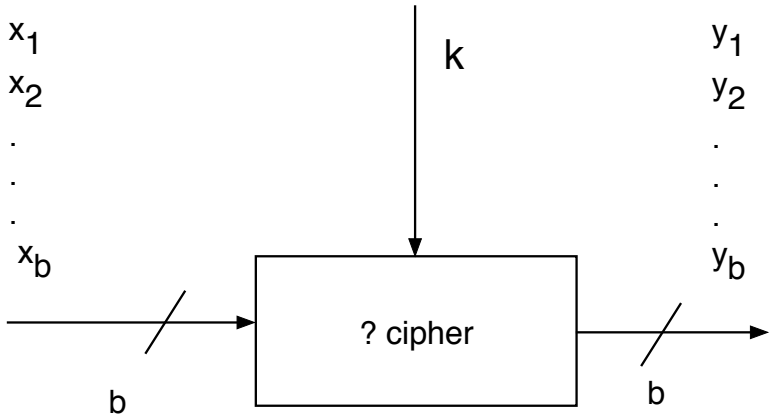
October 2, 2020

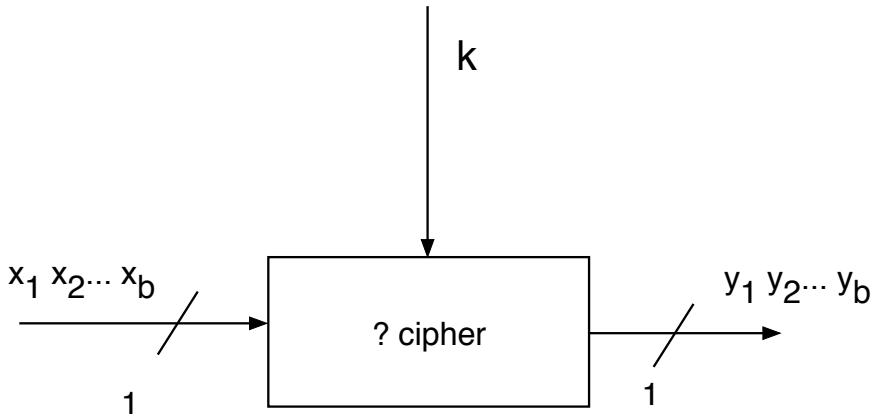
Session 3: Introduction to Cryptography - Symmetric Cryptography

1. Symmetric ciphers
   - Stream ciphers
     - A5/1 cipher
     - RC4 - Keystream generation

Symmetric ciphers

Stream cipher vs. Block cipher





Stream Cipher-How it works

- A key K of n bits is stretched into a long keystream S

$$StreamCipher(K) = S$$

Stream cipher Encryption and Decryption

$x_i, y_i, s_i \in \{0, 1\}$: individual bits of plaintext, ciphertext and keystream

- Encryption: $y_i = x_i + s_i \ (mod\ 2)$
- Decryption: $x_i = y_i + s_i \ (mod\ 2)$

Stream ciphers

- Modulo 2 addition is equivalent to XOR operation
- Plaintext P, Ciphertext C

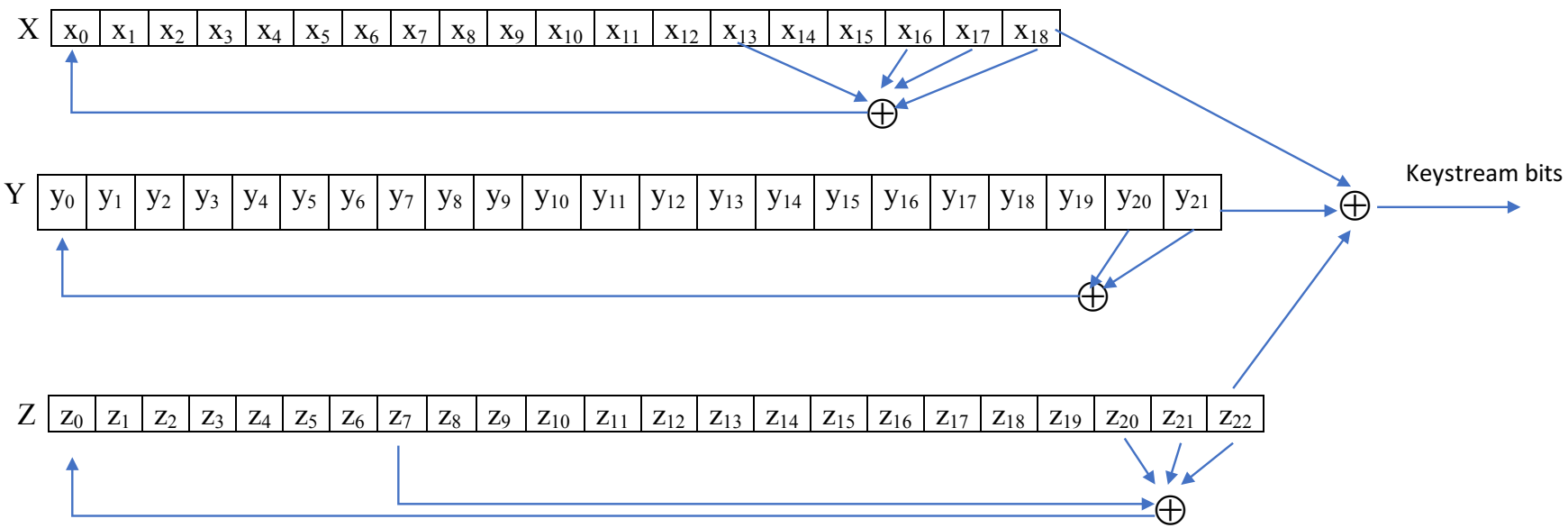
$$C = P \oplus S$$

$$P = C \oplus S$$

A5/1 cipher

- designed for hardware, used in GSM
- Key of 64 bits initially fills in 3 Linear Feedback Shift Registers (LFSRs):
  - X 19 bits
  - Y 22 bits
  - Z 23 bits
- How to obtain every single keystream bit?

A5/1 - Keystream generator



A5/1 - Keystream generator

- LFSR steps:
  - X steps then:

$$x_i = x_{i-1}$$

$$x_0 = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$$

  - Y steps then: ?
  - Z steps then: ?
- A single keystream bit: s = ?

A5/1 - Keystream generation

When they step?

- Majority vote function: $m = maj(x, y, z)$, $m = 0$ when majority of (x,y,z) is 0, otherwise $m = 1$
- In A5/1, $m = maj(x_8, y_{10}, z_{10})$
  - if $x_8 = m$, then X steps
  - if $y_{10} = m$, then Y steps
  - if $z_{10} = m$, then Z steps

RC4 algorithm

- Designed for software, e.g, SSL, WEP...
- Key: N bytes, $1 \leq N \leq 256$
- Lookup table: a 256-byte array S
  - Initialized from the key to the identify permutation: Key-scheduling Algorithm (KSA)
  - can be one of all 256 possible permutations of 256 bytes
- From the lookup table, a keystream byte is generated at each step: Pseudo-random generation algorithm (PRGA)

RC4 - Keystream generation

- KSA pseudo code:

1: **for** $i = 1$ to 255 **do**
2: $S[i] = i$
$K[i] = key[i \ mod \ N]$
3: **end for**
j=0
4: **for** $i = 0$ to 255 **do**
5: $j = (j + S[i] + K[i]) \ mod \ 256$
Swap(S[i], S[j])
6: **end for**
i=j=0

RC4 - Keystream generation

- PRGA pseudo code:
  $i = (i + 1)\ mod\ 256$
  $j = (j + S[i])\ mod\ 256$
  $Swap(S[i], S[j])$
  $t = (S[i] + S[j])\ mod\ 256$
  $KeystreamByte = S[t]$