# Crypto 3 - Aug 30
# Symmetric Key Crypto

# Symmetric Key Crypto

- Stream cipher — like a one-time pad
  - Key is relatively short
  - Key is stretched into a long **keystream**
  - Keystream is then used like a one-time pad
- Block cipher — based on codebook concept
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employ both "confusion" and "diffusion"

# Stream Ciphers

# Stream Ciphers

- Not as popular today as block ciphers
- We'll discuss two examples
- A5/1
  - Based on shift registers
  - Used in GSM mobile phone system
- RC4
  - Based on a changing lookup table
  - Used many places

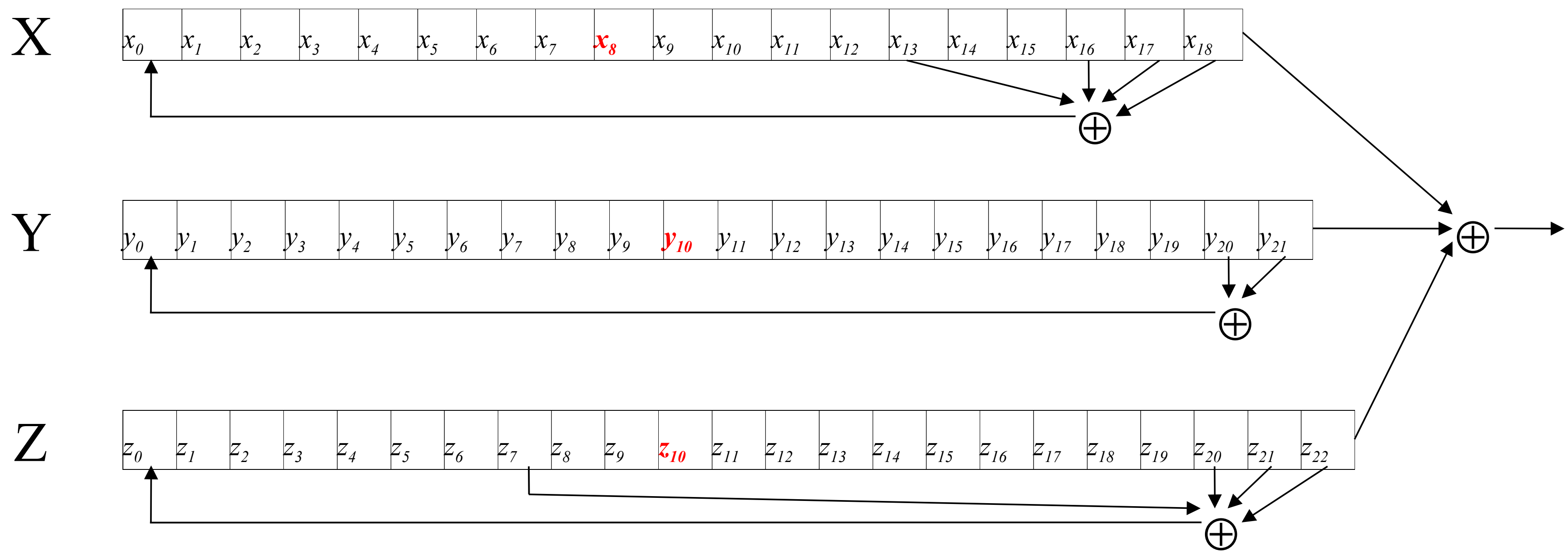
# A5/1

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

# A5/1

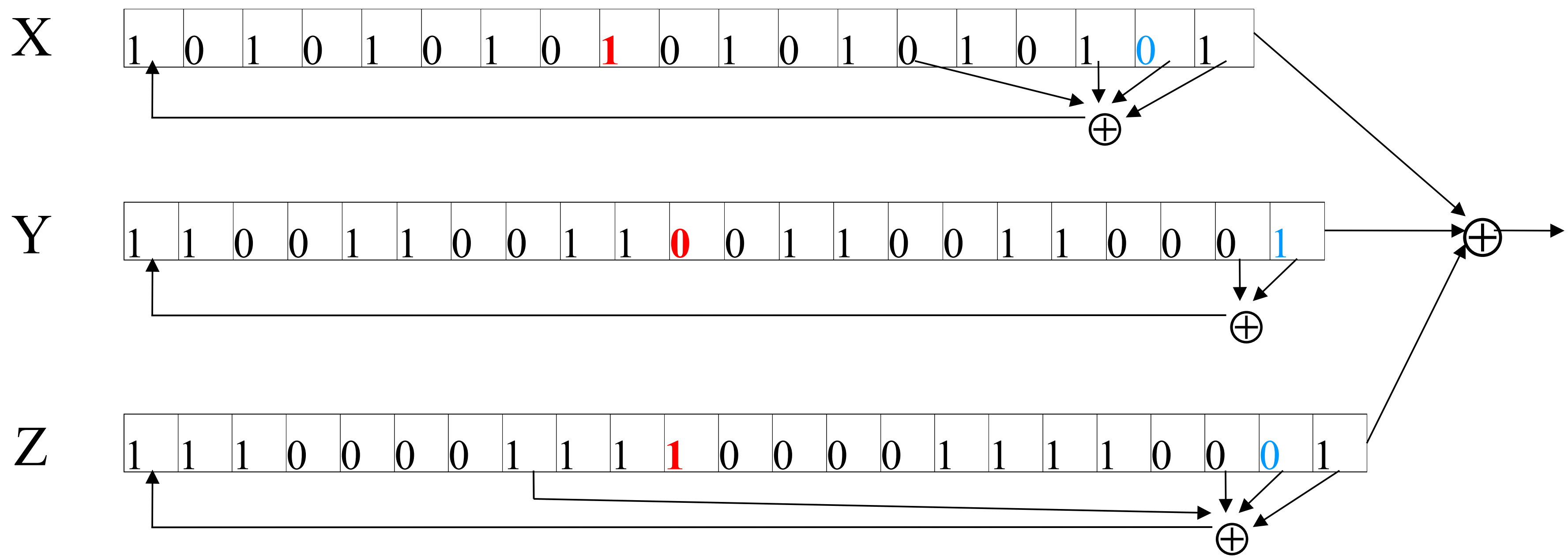
- At each step: $m = \text{maj}(x_8, y_{10}, z_{10})$
  - Examples: $\text{maj}(0,1,0) = 0$ and $\text{maj}(1,1,0) = 1$
- If $x_8 = m$ then X steps
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_i = x_{i-1}$ for $i = 18,17,\ldots,1$ and $x_0 = t$
- If $y_{10} = m$ then Y steps
  -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

# A5/1




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

# A5/1




- In this example, $m = \text{maj}(x_8, y_{10}, z_{10}) = \text{maj}(\mathbf{1},\mathbf{0},\mathbf{1}) = \mathbf{1}$
- Register X steps, Y does not step, and Z steps
- Keystream bit is XOR of right bits of registers
- Here, keystream bit will be $0 \oplus 1 \oplus 0 = 1$

# Shift Register Crypto

- Shift register-based crypto is efficient in hardware
- Harder to implement in software
- In the past, very popular
- Today, more is done in software due to faster processors
- Shift register crypto still used some

# RC4

- A self-modifying lookup table
- Table always contains some permutation of 0,1,…,255
- Initialize the permutation using key
- At each step, RC4
  - Swaps elements in current lookup table
  - Selects a keystream **byte** from table
- Each step of RC4 produces a byte
  - Efficient in software
- Each step of A5/1 produces only a bit
  - Efficient in hardware

# RC4 Initialization

- S[] is permutation of 0,1,...,255
- key[] contains N bytes of key

```
for i = 0 to 255
    S[i] = i
    K[i] = key[i (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 j
i = j = 0
```

# RC4 Keystream

- For each keystream byte, swap table elements 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
- **Note:** first 256 bytes must be discarded
    - Otherwise attacker may be able to recover key

# Stream Ciphers

- Stream ciphers were big in the past
  - Efficient in hardware
  - Speed needed to keep up with voice, etc.
  - Today, processors are fast, so software-based crypto is fast enough
- Future of stream ciphers?
  - Shamir: "the death of stream ciphers"
  - May be exaggerated…

# Block Ciphers

# (Iterated) Block Cipher

- Plaintext and ciphertext consists of fixed sized blocks
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of key and the output of previous round
- Usually implemented in software

# Feistel Cipher

- **Feistel cipher** refers to a type of block cipher design, not a specific cipher
- Split plaintext block into left and right halves: Plaintext = $(L_0,R_0)$
- For each round $i=1,2,\ldots,n$, compute

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus F(R_{i-1},K_i)$

  where $F$ is **round function** and $K_i$ is **subkey**
- Ciphertext = $(L_n,R_n)$

# Feistel Cipher

- Decryption: Ciphertext = $(L_n,R_n)$
- For each round $i=n,n-1,\ldots,1$, compute

  $R_{i-1} = L_i$

  $L_{i-1} = R_i \oplus F(R_{i-1},K_i)$

  where $F$ is round function and $K_i$ is subkey
- Plaintext = $(L_0,R_0)$
- Formula "works" for any function $F$
- But only secure for certain functions $F$

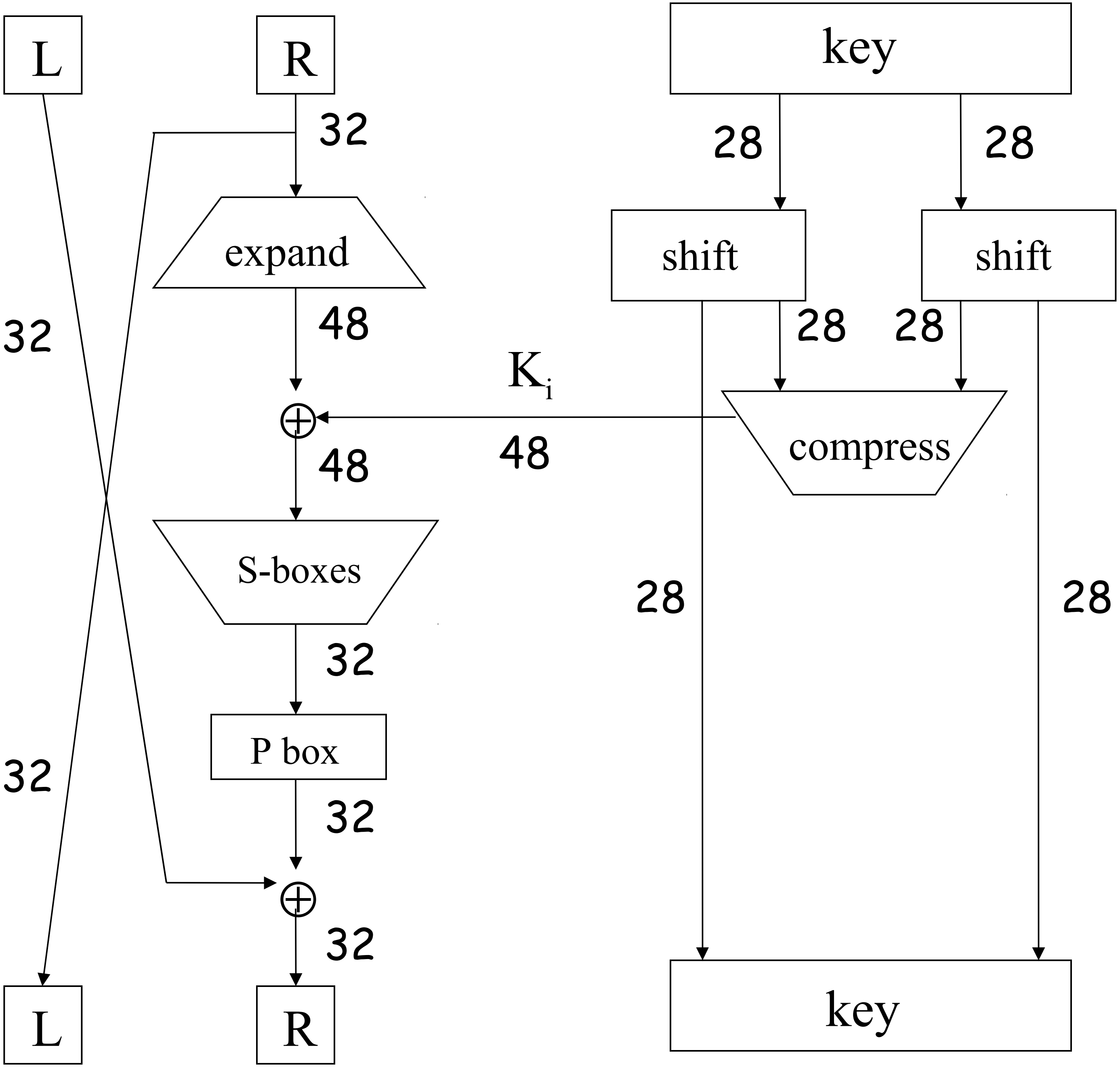
# Data Encryption Standard

- DES developed in 1970's
- Based on IBM Lucifer cipher
- U.S. government standard
- DES development was controversial
  - NSA was secretly involved
  - Design process not open
  - Key length was reduced
  - Subtle changes to Lucifer algorithm

# DES Numerology

- DES is a Feistel cipher
  - 64 bit block length
  - 56 bit key length
  - 16 rounds
  - 48 bits of key used each round (subkey)
- Each round is simple (for a block cipher)
- Security depends primarily on "S-boxes"
  - Each S-boxes maps 6 bits to 4 bits

# One Round of DES



# DES Expansion Permutation

- Input 32 bits

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

- Output 48 bits

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 0 | 1 | 2 | 3 | 4 | 3 | 4 | 5 | 6 | 7 | 8 |
| 7 | 8 | 9 | 10 | 11 | 12 | 11 | 12 | 13 | 14 | 15 | 16 |
| 15 | 16 | 17 | 18 | 19 | 20 | 19 | 20 | 21 | 22 | 23 | 24 |
| 23 | 24 | 25 | 26 | 27 | 28 | 27 | 28 | 29 | 30 | 31 | 0 |

# DES S-box

- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- S-box number 1

input bits (0,5)

↓ input bits (1,2,3,4)

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 1110 | 0100 | 1101 | 0001 | 0010 | 1111 | 1011 | 1000 | 0011 | 1010 | 0110 | 1100 | 0101 | 1001 | 0000 | 0111 |
| 01 | 0000 | 1111 | 0111 | 0100 | 1110 | 0010 | 1101 | 0001 | 1010 | 0110 | 1100 | 1011 | 1001 | 0101 | 0011 | 1000 |
| 10 | 0100 | 0001 | 1110 | 1000 | 1101 | 0110 | 0010 | 1011 | 1111 | 1100 | 1001 | 0111 | 0011 | 1010 | 0101 | 0000 |
| 11 | 1111 | 1100 | 1000 | 0010 | 0100 | 1001 | 0001 | 0111 | 0101 | 1011 | 0011 | 1110 | 1010 | 0000 | 0110 | 1101 |

# DES P-box

- Input 32 bits

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

- Output 32 bits

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 6 | 19 | 20 | 28 | 11 | 27 | 16 | 0 | 14 | 22 | 25 | 4 | 17 | 30 | 9 |
| 1 | 7 | 23 | 13 | 31 | 26 | 2 | 8 | 18 | 12 | 29 | 5 | 21 | 10 | 3 | 24 |

# DES Subkey

- 56 bit DES key, numbered 0,1,2,…,55
- Left half key bits, LK

```
49 42 35 28 21 14  7
 0 50 43 36 29 22 15
 8  1 51 44 37 30 23
16  9  2 52 45 38 31
```

- Right half key bits, RK

```
55 48 41 34 27 20 13
 6 54 47 40 33 26 19
12  5 53 46 39 32 25
18 11  4 24 17 10  3
```

# DES Subkey

- For rounds `i=1,2,...,16`
  - Let LK = (LK circular shift left by $r_i$)
  - Let RK = (RK circular shift left by $r_i$)
  - Left half of subkey $K_i$ is of LK bits

```
13 16 10 23  0  4  2 27 14  5 20  9
22 18 11  3 25  7 15  6 26 19 12  1
```

  - Right half of subkey $K_i$ is RK bits

```
12 23  2  8 18 26  1 11 22 16  4 19
15 20 10 27  5 24 17 13 21  7  0  3
```

# DES Subkey

- For rounds 1, 2, 9 and 16 the shift $r_i$ is 1, and in all other rounds $r_i$ is 2
- Bits 8,17,21,24 of LK omitted each round
- Bits 6,9,14,25 of RK omitted each round
- **Compression permutation** yields 48 bit subkey $K_i$ from 56 bits of LK and RK
- **Key schedule** generates subkey

# DES Last Word (Almost)

- An initial perm P before round 1
- Halves are swapped after last round
- A final permutation (inverse of P) is applied to $(R_{16}, L_{16})$ to yield ciphertext
- None of these serve any security purpose

# Security of DES

- Security of DES depends a lot on S-boxes
  - Everything else in DES is linear
- Thirty years of intense analysis has revealed no “back door”
- Attacks today use exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were ahead of their time