Chapter 6 – Contemporary Symmetric Ciphers

Triple DES

- A replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Before AES alternative
  - use multiple encryptions with DES
- Triple-DES is the chosen form

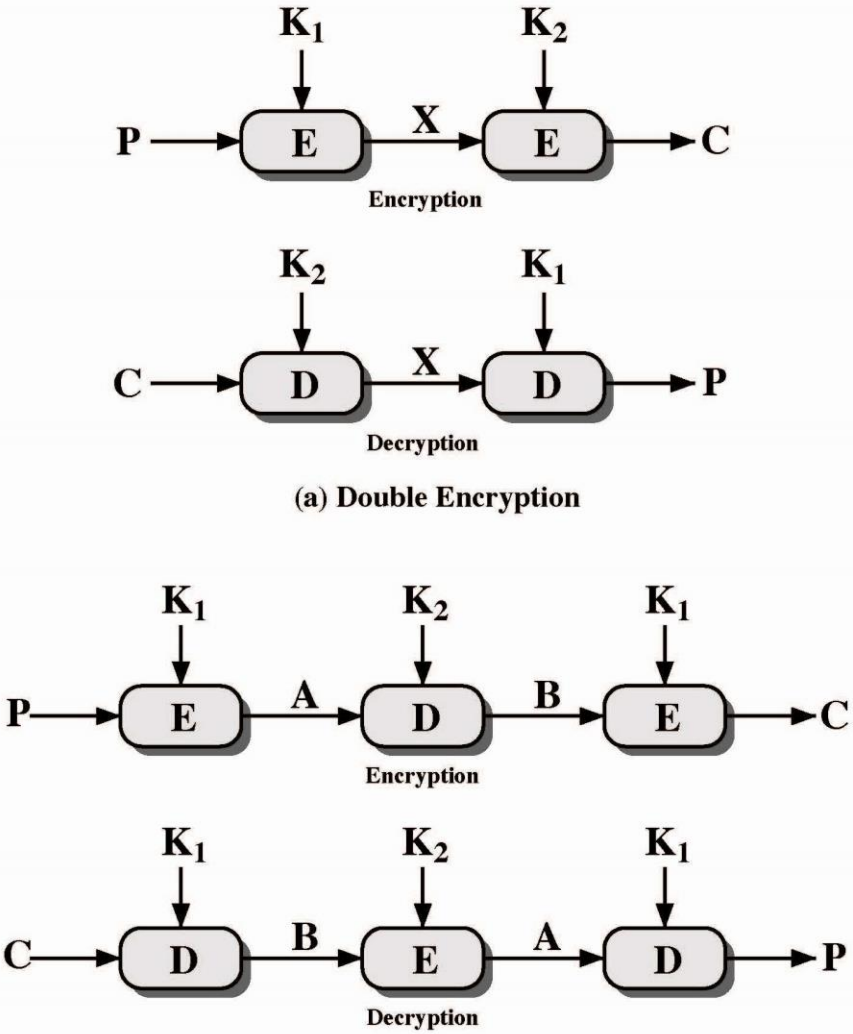

Figure 6.1 Multiple Encryption

Why Triple-DES?

- why not Double-DES?
  - NOT same as some other single-DES use, but have
- meet-in-the-middle attack
  - works whenever use a cipher twice
  - since $X = E_{K1}[P] = D_{K2}[C]$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
  - Key of 56 X 3 = 168 bits seems too long
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - No cryptographic significance to the use of D in the second step
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - some are now adopting Triple-DES with three keys for greater security

Triple-DES with Three-Keys

- although are no practical attacks on two-key Triple-DES have some indications
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- has been adopted by some Internet applications

Blowfish

- a symmetric block cipher designed by Bruce Schneier in 1993/94
- characteristics
  - fast implementation on 32-bit CPUs, 18 clock cycles per byte
  - compact in use of memory, less than 5KB
  - simple structure for analysis/implementation
  - variable security by varying key size
    - Allows tuning for speed/security tradeoff

Blowfish Key Schedule

- uses a 32 to 448 bit key
- used to generate
  - 18 32-bit subkeys stored in P-array: P1 to P18
  - S-boxes stored in $S_{i,j}$,
    - $i=1..4$
    - $j=0..255$

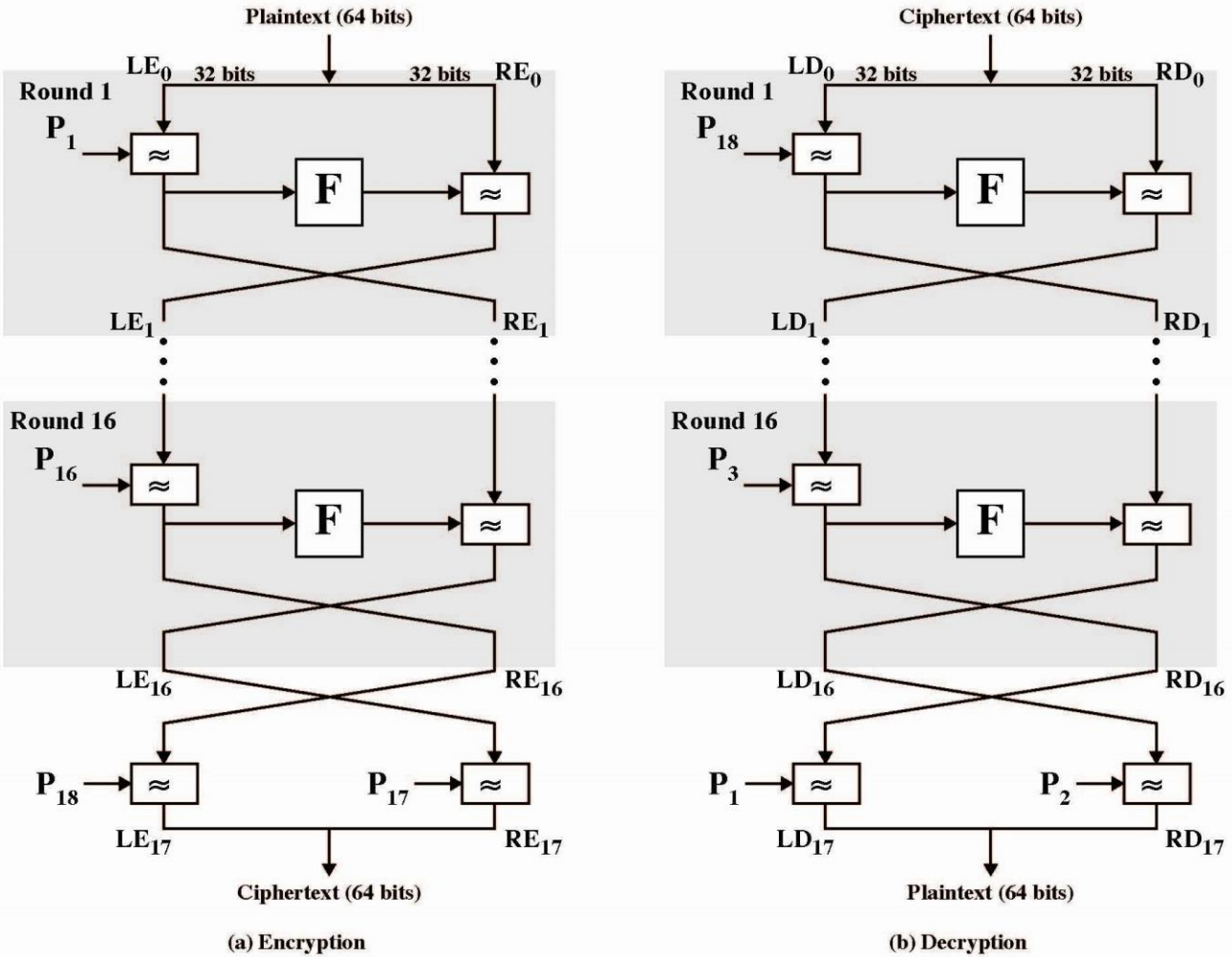

Figure 6.3 Blowfish Encryption and Decryption

Blowfish Encryption

- uses two primitives: addition & XOR
- data is divided into two 32-bit halves L_0 & R_0

for i = 1 to 16 do

$R_i = L_{i-1}$ XOR P_i;

L_i = F[R_i] XOR R_{i-1};

$L_{17} = R_{16}$ XOR P_{18};

$R_{17} = L_{16}$ XOR i_{17};

- where

F[a,b,c,d] = (($S_{1,a}$ + $S_{2,b}$) XOR $S_{3,c}$) + $S_{4,a}$

Break 32-bit R_i into (a,b,c,d)

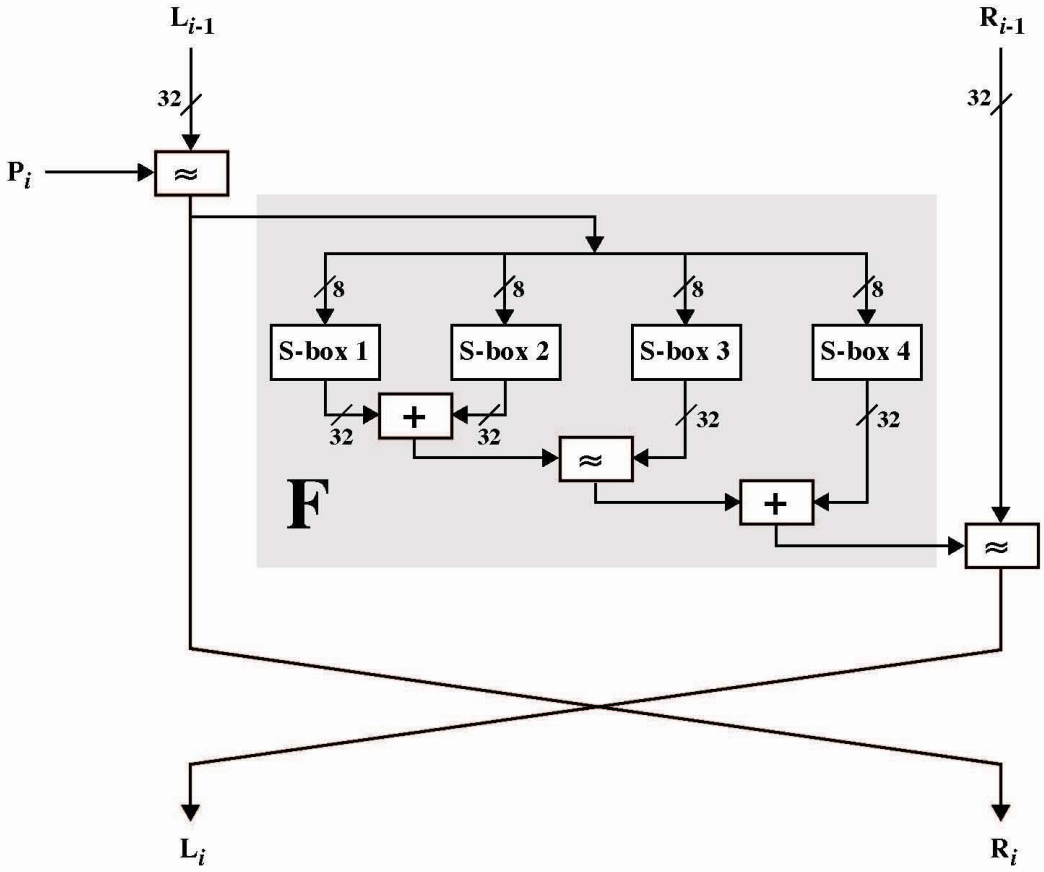

Figure 6.4 Detail of Single Blowfish Round

Discussion

- provided key is large enough, brute-force key search is not practical, especially given the high key schedule cost
- key dependent S-boxes and subkeys make analysis very difficult
  - Very few cryptoanalysis results on blowfish
- changing both halves in each round increases security
  - Some study shows improved avalanche effects

RC5

- can vary key size / input data size / #rounds
- very clean and simple design
- easy implementation on various CPUs
- yet still regarded as secure
  - Vary parameters to achieve tradeoffs

RC5 Ciphers

- RC5 is a family of ciphers RC5-w/r/b
  - w = word size in bits (16/32/64) data=2w
  - r = number of rounds (0..255)
  - b = number of bytes in key (0..255)
- nominal version is RC5-32/12/16
  - ie 32-bit words so encrypts 64-bit data blocks
  - using 12 rounds
  - with 16 bytes (128-bit) secret key

RC5 Key Expansion

- RC5 uses 2r+2 subkey words (w-bits)
  - Two subkeys for each round
  - 2 subkeys for additional operations
- subkeys are stored in array `S[i]`, i=0..t-1
- Key expansion: fill in pseudo-random bits to the original key K
- Certain amount of *one-wayness*
  - Difficult to determine K from S

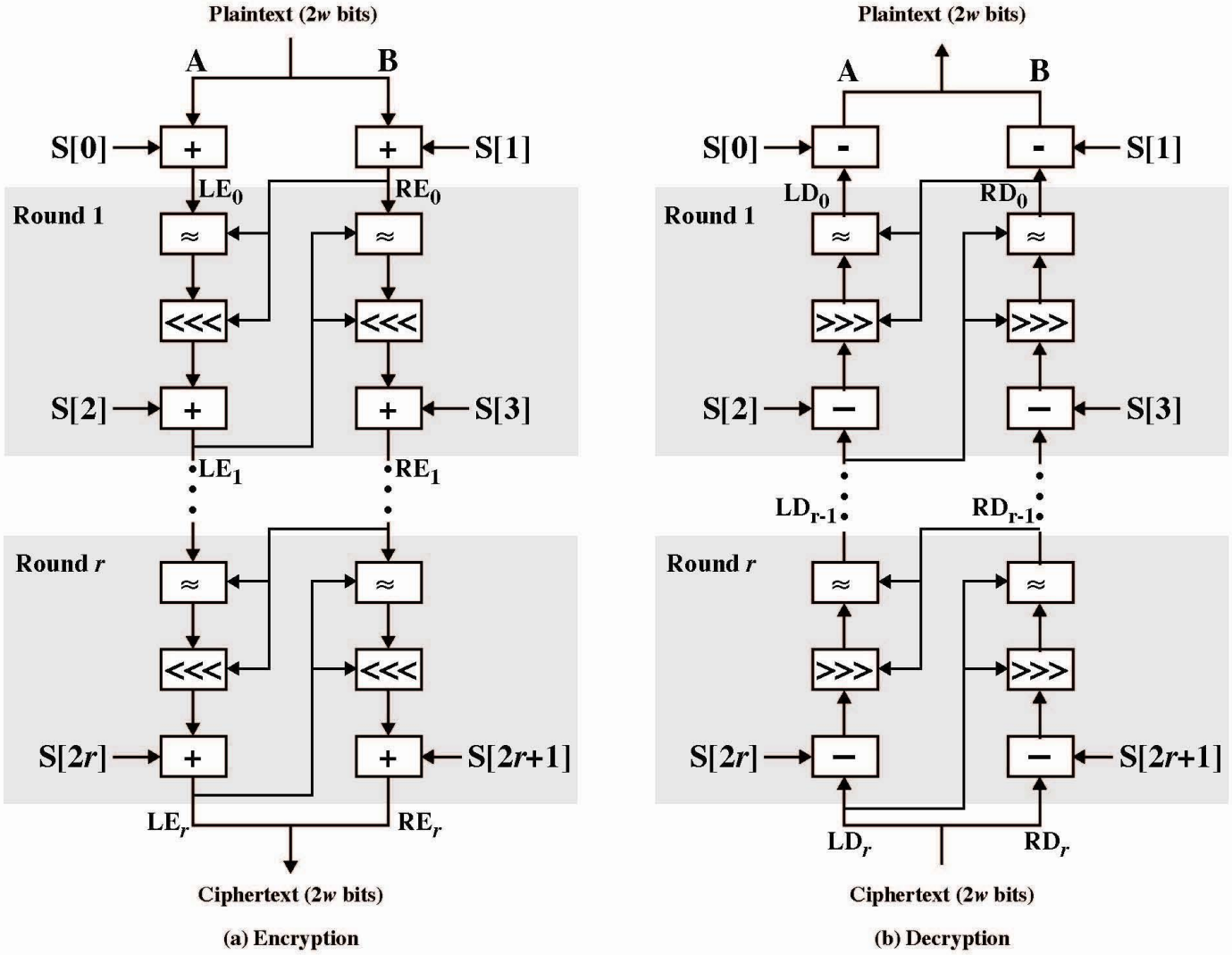

Figure 6.6 RC5 Encryption and Decryption

RC5 Encryption

- split input into two halves A & B

  $L_0 = A + S[0];$

  $R_0 = B + S[1];$

  for i = 1 to r do

  $L_i = ((L_{i-1} \text{ XOR } R_{i-1}) <<< R_{i-1}) + S[2 \times i];$

  $R_i = ((R_{i-1} \text{ XOR } L_i) <<< L_i) + S[2 \times i + 1];$
- each round is like 2 DES rounds
- note rotation is main source of non-linearity
- need reasonable number of rounds (eg 12-16)
- Striking features: simplicity, data-dependent rotations

RC5 Modes

- RFC2040 defines 4 modes used by RC5
  - RC5 Block Cipher, is ECB mode
  - RC5-CBC, input length is a multiples of 2w
  - RC5-CBC-PAD, any length CBC with padding
    - Output can be longer than input
  - RC5-CTS, CBC with padding
    - Output has same length than input

Block Cipher Characteristics

- features seen in modern block ciphers are:
  - variable key length / block size / no rounds
  - mixed operators
    - data/key dependent rotation
    - key dependent S-boxes
  - more complex key scheduling
    - Lengthy key generation, simple encryption rounds
  - operation of full data in each round

Stream Ciphers

- process the message bit by bit (as a stream)
- typically have a (pseudo) random **key stream**
- combined (XOR) with plaintext bit by bit
- randomness of **key stream** completely destroys any statistically properties in the message
  - $C_i = M_i$ XOR $StreamKey_i$
- what could be simpler!!!!
- but must never reuse key stream
  - otherwise can remove effect and recover messages

Block/Stream Ciphers

- Stream ciphers
  - For applications that require encryt/decryt of a stream of data
  - Examples: data communication channel, brower/web link
- Block ciphers
  - For applications dealing with blocks of data
  - Examples: file transfer, e-mail, database
- Either type can be used in virtually any application

Stream Cipher Properties

- some design considerations are:
  - long period with no repetitions
  - statistically random
  - Highly nonlinear correlation

RC4

- variable key size, byte-oriented stream cipher
- widely used (web SSL/TLS between browser and server, wireless WEP)
- key forms random permutation of a 8-bit string
- uses that permutation to scramble input info processed a byte at a time

RC4 Security

- claimed secure against known attacks
  - have some analyses, none practical
- result is very non-linear
- since RC4 is a stream cipher, must **never reuse a key**

Summary

- have considered:
  - some other modern symmetric block ciphers
  - Triple-DES
  - Blowfish
  - RC5
  - briefly introduced stream ciphers