Feistel Ciphers and DES

Dr. Demetrios Glinos
University of Central Florida

CIS3360 - Security in Computing

Readings

- "Computer Security: Principles and Practice", 3rd Edition, by William Stallings and Lawrie Brown
  - Sec. 20.2

For More Information

- There are many online resources on Feistel Ciphers and DES, including:

- Wikipedia page on Feistel Ciphers
  http://en.wikipedia.org/wiki/Feistel_cipher

- Wikipedia page on DES
  http://en.wikipedia.org/wiki/Data_Encryption_Standard

Outline

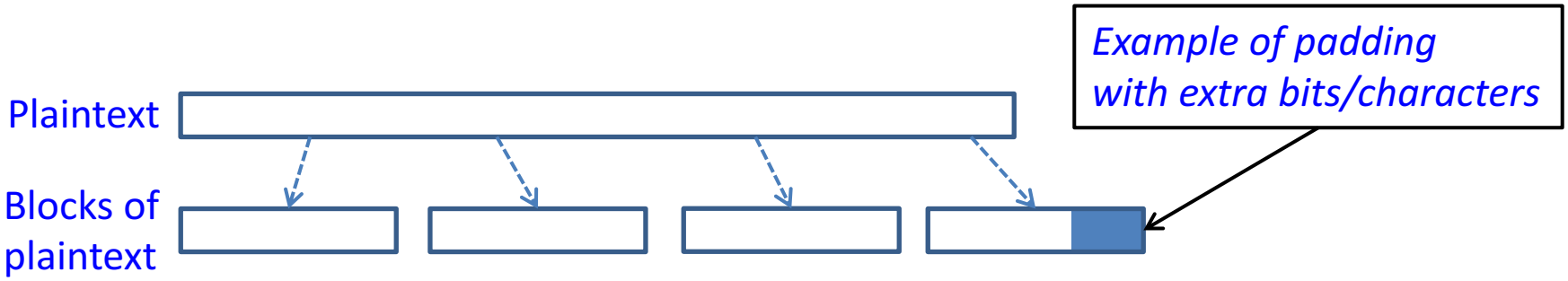
- Block Cipher Review
- Substitution-Permutation Networks
- Diffusion and Confusion
- Feistel Cipher Structure
- DES and Triple-DES

Block Cipher Review (1)

- Modern symmetric ciphers typically operate on blocks of data
  - Similar to the "blocking" we have seen for
    - Playfair cipher – block size is 2
    - Hill cipher – block size is size of vector determined by key matrix
    - Other: for example, Program #1 using Vigenere as block cipher
- Block ciphers are widely used
  - Blocks used are typically much larger than classical single or double-character ciphers (8 or 16 bits)
  - Typical block sizes: 64, 128, and 256 bits
- We will examine DES (this lecture) and AES (next lecture) to illustrate modern block cipher design principles

Block Cipher Review (2)

- In a **block cipher:**
  - Plaintext and ciphertext are processed in blocks of fixed length **b** (e.g., 128 bits)
  - A plaintext of length **n** is partitioned into a sequence of **m blocks**, P[0], …, P[m−1], where $\mathbf{n} \leq \mathbf{bm} < \mathbf{n + b}$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks
  - All blocks are the same size
  - Last block may need include some padding (if run out of plaintext)



Substitution-Permutation Networks

- **Claude Shannon** introduced idea of ***substitution-permutation (S-P) networks*** in 1949 paper
- S-P nets form the basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations we have studied before:
  - ***substitution*** **(S-box)**
  - ***permutation*** **(P-box)**
- S-P nets provide ***confusion*** and ***diffusion*** for message and key

Diffusion and Confusion

- ***Goal of cipher design is to completely obscure statistical properties of original message***
- Shannon's idea: combining S & P elements to obtain:
- **Diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
  - *i.e., changing one plaintext bit must affect as many ciphertext bits as possible*
- **Confusion** – makes relationship between ciphertext and key as complex as possible
  - *i.e., changing one bit of the encryption key must affect as many ciphertext bits as possible*

Feistel Cipher Structure

- Developed by **Horst Feistel** at IBM in early 70s and first implemented in the Lucifer cipher for Lloyd's of London monetary transactions
- Based on concept of ***invertible product cipher*** (i.e, the ability to reverse a sequence of cryptographic transformations)
- Implements Shannon's S-P net concept
- Involves multiple ***rounds*** of encryption/decryption
- Uses the ***same cipher algorithm*** for both encryption and decryption
- But uses ***multiple keys (subkeys)***, which are different (*so, the order matters*)

Feistel Cipher Operation

- Each ***round*** of encryption involves a ***substitution step***, followed by a ***permutation step***
- **Here's how it works:**
  1. Plaintext input block is first split into 2 halves, labelled right (R) and left (L)
  2. In each round:
     a) R passes through unchanged (and after step (c) below becomes the L half for the next round)
     b) L goes through a ***substitution step*** that uses a function (known as the ***Feistel function***) that ***depends on R and the subkey for the round***
     c) Then ***permute*** by swapping halves

Feistel Cipher Computations

Given plaintext block **P**, we divide it into **L_0** and **R_0** halves

For Round 1:

$$L_1 = R_0$$
$$R_1 = L_0 \oplus F(R_0, K_1)$$

•••

For Round i:

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

NOTE 1: we use of the XOR operator ⊕

NOTE 2: F(R,K) is the Feistel function; value depends on current R and K

In other words, for example:

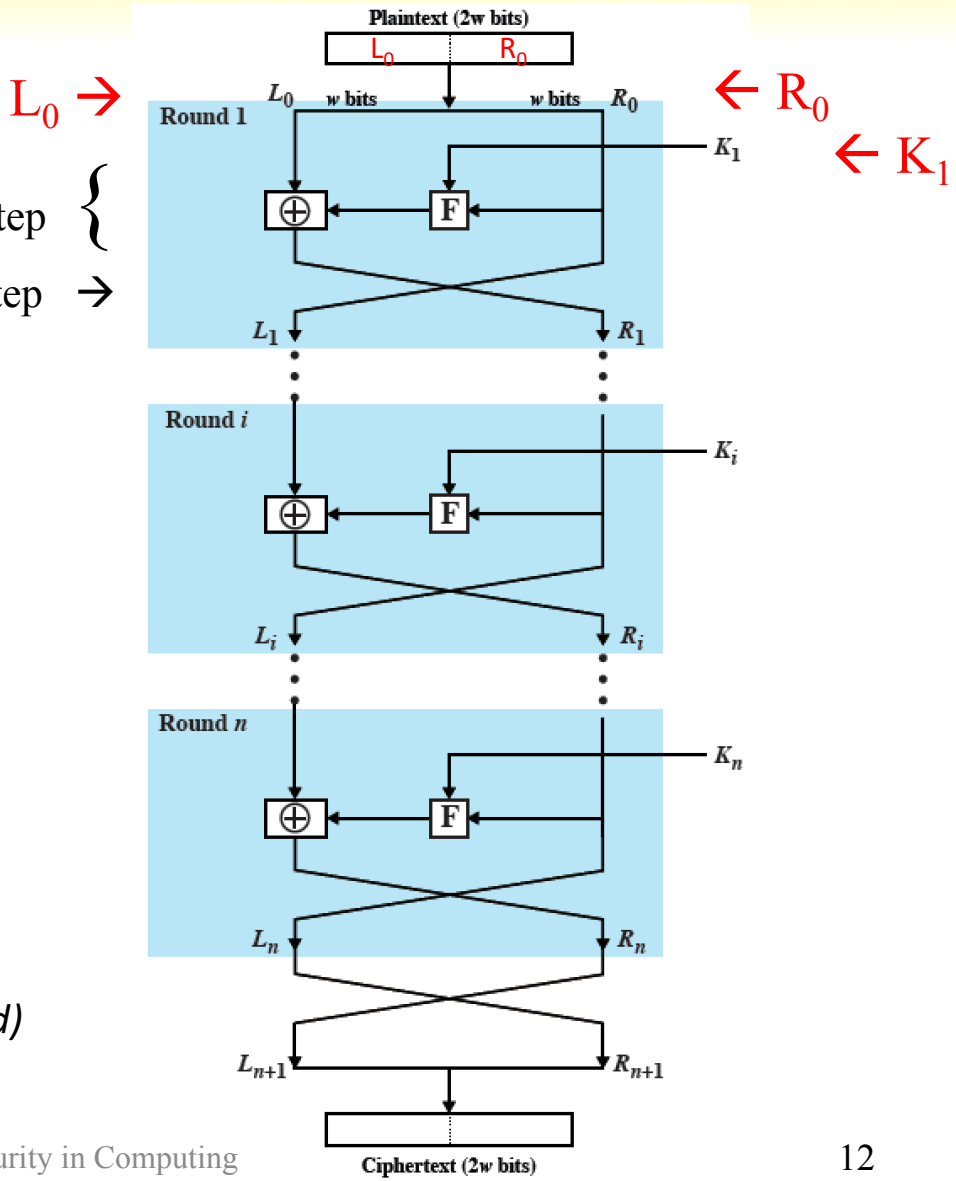
Round 7 uses the L and R produced by Round 6

The L produced by Round 7 is just the R produced by Round 6

The R produced by Round 7 is computed per formula above

Feistel Cipher Round Structure

This is just one block of many →

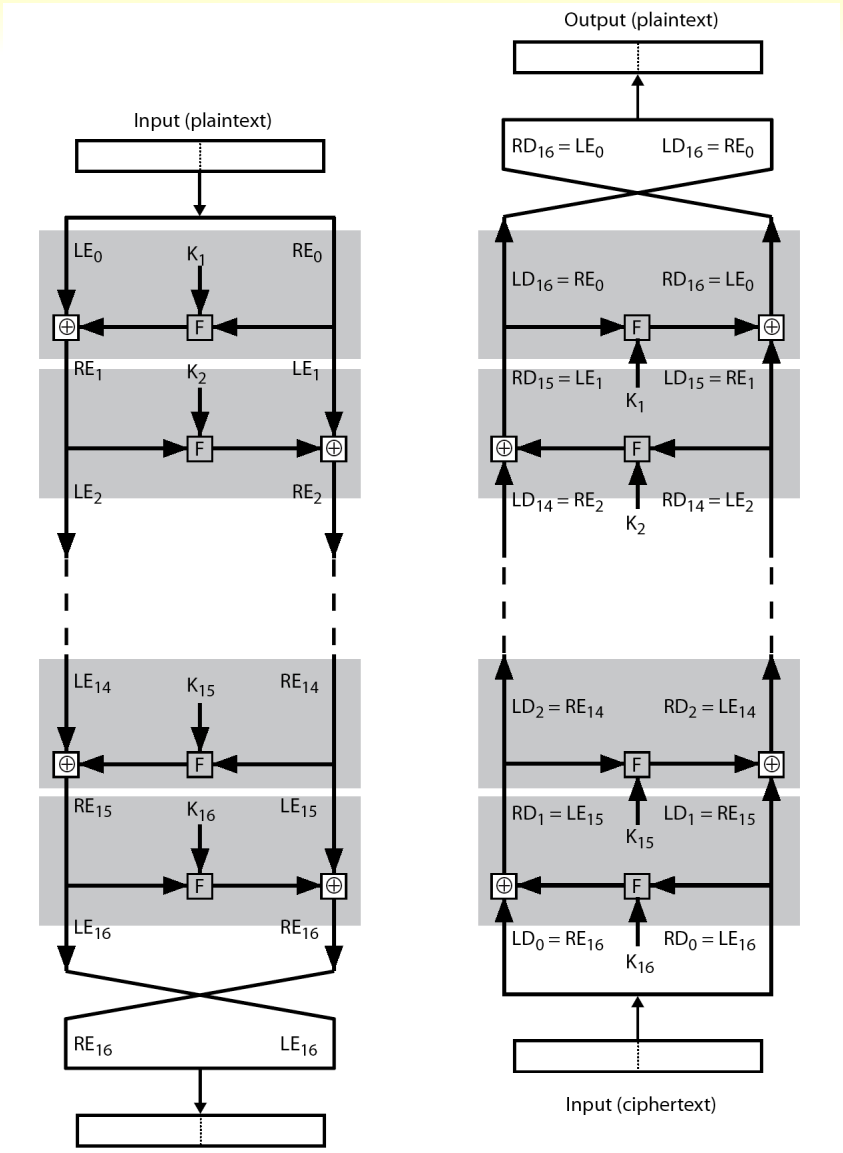


Notes:

- *There is a different key for each round (called a "subkey")*
- *The Feistel function "F" can be any encryption algorithm*

source: Fig. 20.1 (annotated)

Feistel Cipher Encryption & Decryption



NOTE: To decrypt, we perform the same operations, but we apply the keys in the reverse order

source: Supplemental Text (annotated)

Data Encryption Standard (DES)

- The most widely used block cipher in world
- Adopted in 1977 by NBS (now NIST) as FIPS (Federal Information Processing Standard) PUB 46
- Based on Lucifer cipher developed by IBM for Lloyds of London (for cash transfers)
- Has been considerable controversy over its security
- Code cracked in 1999 by the Electronics Frontiers Foundation using special hardware
- Following which NIST directed use of “Triple DES”
- Ultimately replaced by AES for Federal use, but Triple DES is still in wide use commercially.

DES Processing

- Uses ***16-round*** Feistel structure
- Encrypts ***64-bit data*** using ***56-bit key***
- ***The subkeys for the rounds are generated from the single input encryption key***
- Complicated Feistel function consisting of:
    1. An expansion permutation ("E-step")
    2. XOR'ing with the round key ("key mixing")
    3. Then performing a substitution with 8 "S-boxes"
    4. And finally a permutation with a "P-box"
- Feistel function result XOR'd with Left half, as usual
- Left and Right results permuted by swapping, as usual

DES Round Structure (simplified)

- splits a 64-bit input block into two 32-bit L & R halves
- as for any Feistel cipher can describe as:

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

- Feistel function **F** takes 32-bit R half and **48-bit subkey**:
  - *expands* R to 48-bits using E-step permutation
  - *applies* subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P-box

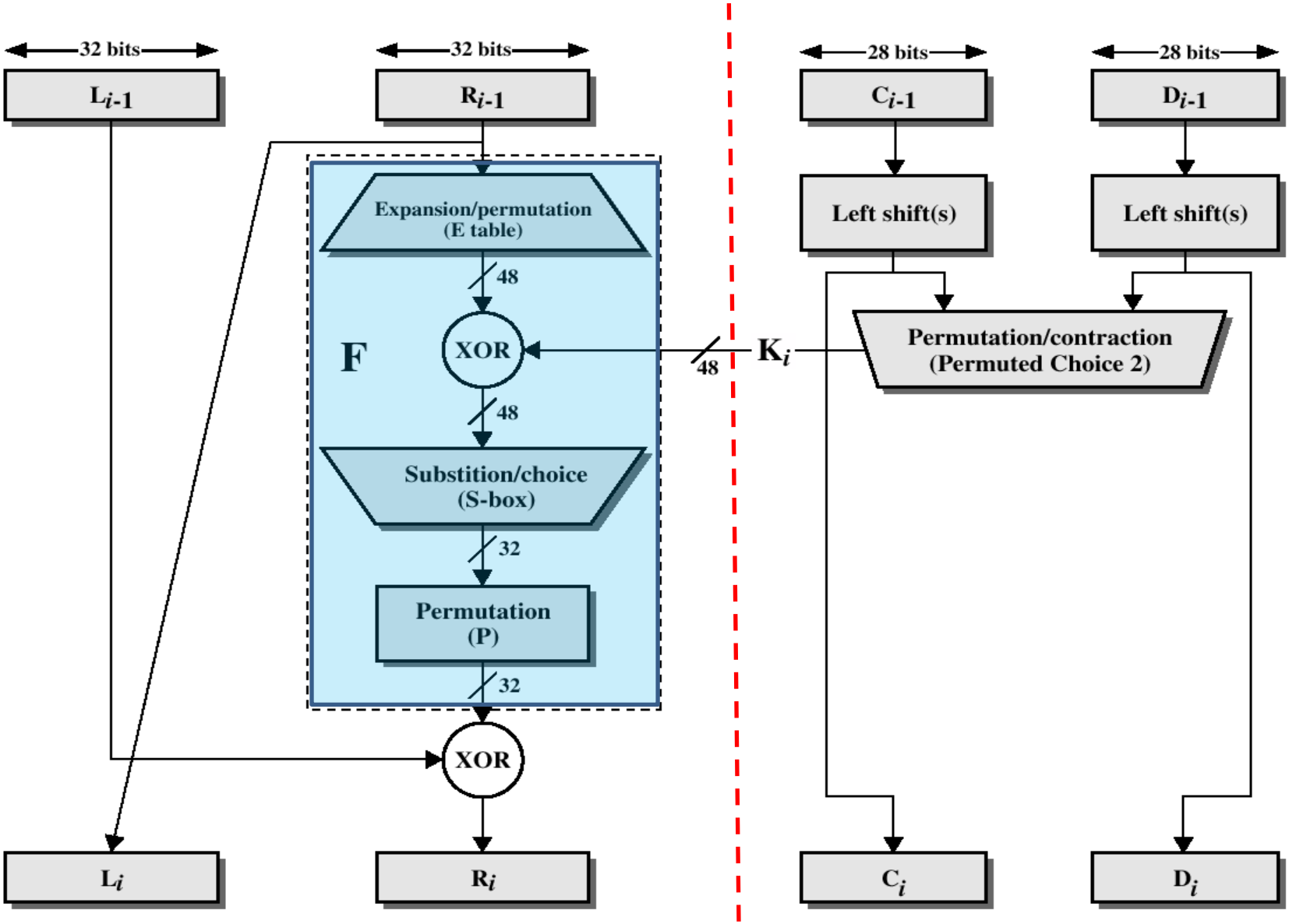


Figure 2.4 Single Round of DES Algorithm

source: Supplemental Text (annotated)

DES Expansion Permutation (E-step)

- ***Expands 32-bit Right half to 48 bits***
- Performed as follows:

1. Divide 32-bit block into **eight** 4-bit words
2. *Add a bit to the left* of each 4-bit word that is the last bit of the ***previous*** 4-bit word (with wraparound)
3. *Add a bit to the right* of each 4-bit word that is the first bit of the ***next*** 4-bit word (with wraparound)
4. Result is **eight** 6-bit words which are combined into a single 48-bit block

DES Substitution Step

- ***Takes 48-bit input and returns 32-bit output***
- Performed as follows:
  - Divide 48-bit block into eight 6-bit words
  - Each 6-bit word is fed into a separate S-box and produces a 4-bit output
  - Each S-box has 4 rows and 16 columns and contains numbers in the range 0, 1, 2, …, 15.
  - First and last bit of 6-bit word index the row
  - The middle 4 bits index the column
  - The result is a 4-bit unsigned binary number

S-boxes S_1 & S_2

S1	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
R0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
R1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
R2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
R3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
R0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
R1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
R2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
R3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-Box Example

- To show how S-boxes work, let's just look at 12 bits out of the 48-bit block
  - Suppose we want: the S-Box output for the input FOE_{16}
  - $FOE_{16} = 1111\ 0000\ 1110_2$
  - Input to S_1 = higher order six bits = 111100_2
  - So, the row selector = 10_2 = row 2_{10}
  - And the column selector = 1110_2 = column 14_{10}
  - Thus, the output from $S_1 = 5_{10} = 5_{16}$
  - Input to S_2 = lower order six bits = 001110_2
  - So, the row selector = 00_2 = row 0_{10}
  - And the column selector = 0111_2 = column 7_{10}
  - Output from $S_2 = 4_{10} = 4_{16}$
- Hence, the combined output of S_1 and S_2 is 54_{16}

We perform similar processing of the next 12 bits through S_3 & S_4, the next 12 bits through S_5 & S_6, and the last 12 bits through S_7 & S_8 to complete the processing for the 48-bit block

DES Permutation Step

- **Takes 32-bit input and returns 32-bit output**
- A ***table-lookup fixed permutation*** of the input bits
- Order is:

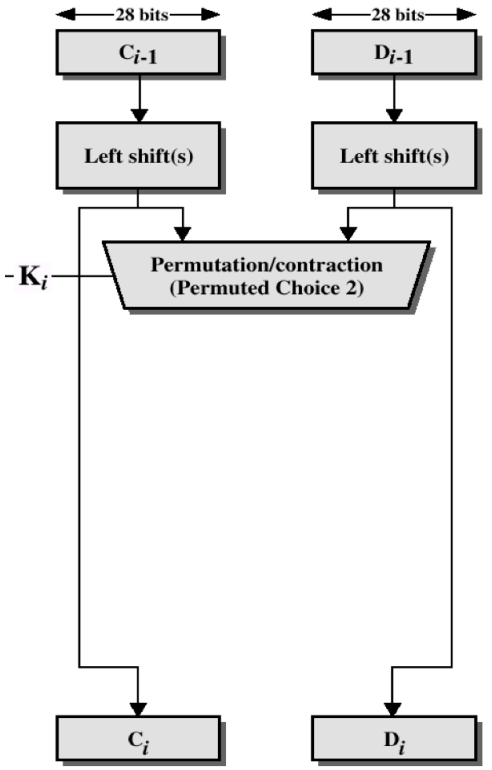
  16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,
  2,8,24,14,32,27,3,9,19,13,30,6,22,11,4,25

  → First bit of output is 16th input bit, 2nd bit of output is 7th bit of input, etc.

  Note: You do NOT need to memorize or write down this permutation order

DES Key Schedule

- Initial key is **56 bits (7 bytes)**
  - Can also be 8 bytes, with last bit of each byte used for parity (so only 56 bit are used)
- Before any round keys are generated, the 56 usable bits are permuted using a table called "***Permutation Choice 1***"
- Thereafter, at the beginning of **each round**:
  - Split 56-bit key block into two 28-bit halves
  - Circularly left shift each block by 1 or 2 bits (1 bit for rounds 1,2,9, and 16; 2 bits for others)
  - Join the shifted halves together and contract to 48 bits using "*Permutation Choice 2*" that selects a fixed 48-bit subset of the 56 bits in a particular order (***this is the subkey for the round***)
- ***NOTE: the circularly shifted 28-bit halves are joined and passed on to the next round to generate the next key***



source: Supplemental Text (annotated)

What Makes DES Strong

- **Substitution step performs diffusion well:**
  - On average, changing 1 bit of 64-bit input block changes 34 bits of the ciphertext block
- **Key schedule performs confusion well:**
  - On average, changing 1 bit of 56-bit encryption key changes 35 bits of the ciphertext block
- **Good size key space: $2^{56} \approx 7.2 \times 10^{16} = 72{,}000{,}000{,}000{,}000{,}000$**
  - Brute force attack at 1,000 keys/microsecond would need about 13 months to test half the keys
  - EFF took 3 days on special hardware
  - If can parallel process 1 million keys at once, can do in about 10 hours
  - This is why we now use 3DES

Triple-DES with Two/Three-Keys

- Single DES is strong, but **Triple DES ("3DES")** is much stronger
  - *at cost of 3 times slower to run*
  - The reason: must encrypt 3 times

- but can use 2 keys with E-D-E sequence
  - **$C = E_{K1}(D_{K2}(E_{K1}(P)))$** and **$P = D_{K1}(E_{K2}(D_{K1}(C)))$**
  - NOTE: encrypt & decrypt equivalent in security
  - if K1=K2 then equivalent to single DES
  - key space ≈ 2^{112}

- can also use Triple-DES with Three-Keys
  - **$C = E_{K3}(D_{K2}(E_{K1}(P)))$** and **$P = D_{K1}(E_{K2}(D_{K3}(C)))$**
  - key space ≈ 2^{168}

- no current known practical attacks for either version
  - but use AES for applications that need better performane