DES implementation

四資工三甲 B10730027 曾瑄翎

- **Converters**

  hex_toBinary
  dec_toBinary
  bin_toHex
  ascii_toBinary

  二進制、十進制、十六進制、ASCII code之間的轉換，注意不省略leading zero

- **Lookup Tables for DES**

  initial_perm
  expansion
  per
  sbox
  final_perm
  pc1
  shift_table
  pc2

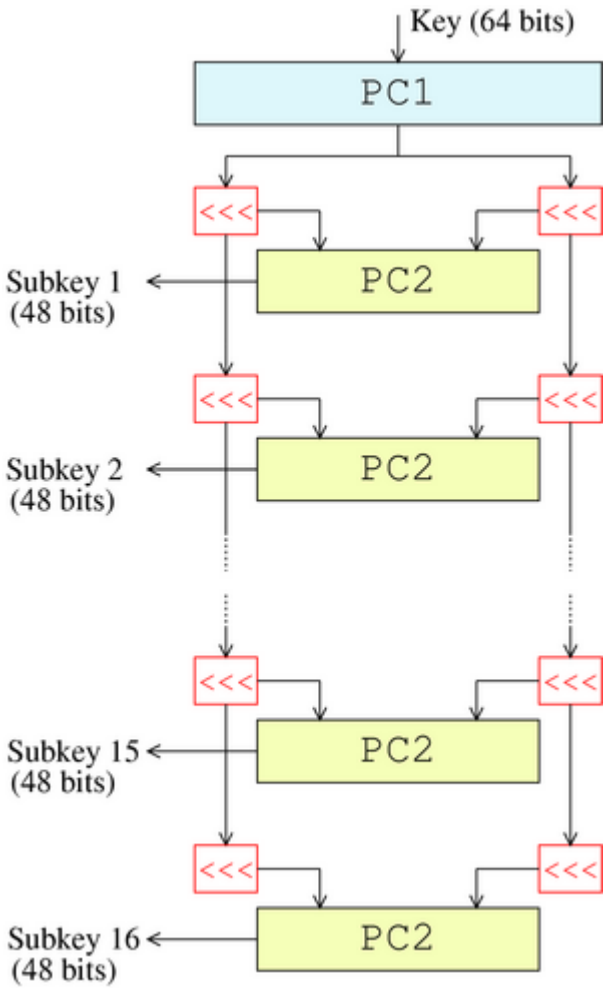
  紀錄DES的各種置換、擴張、移位規則

- **Functions**

  permute
  shift_left
  xor

  實作加密時會用到的函式

- **Key Generation**



如上圖，完成步驟：

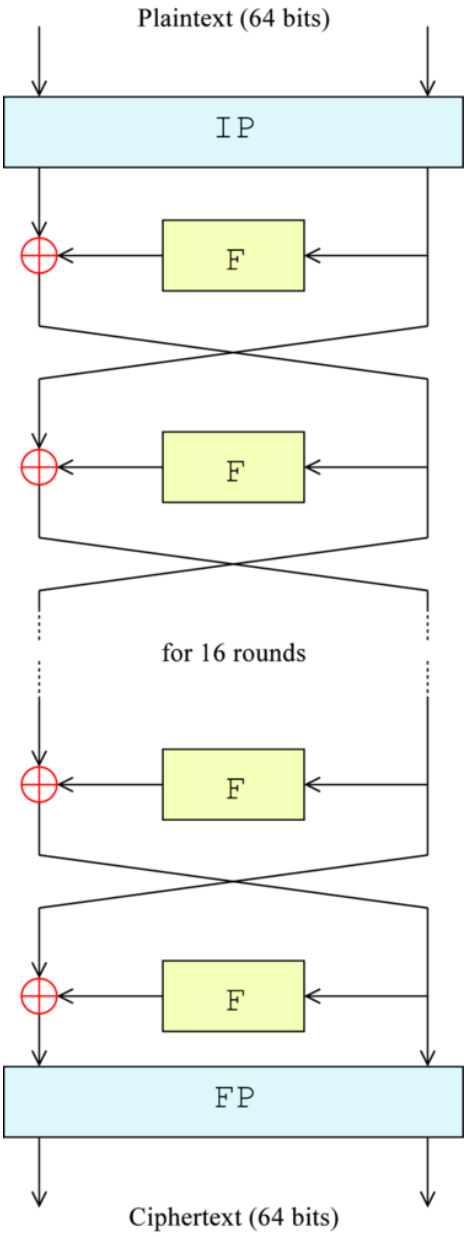
一、使用PC-1，從64位元輸入金鑰中選出56位的金鑰

二、56位分成兩個28位元的半金鑰

三、十六回合：兩個半金鑰都被左移1或2位（查找shift_table），並通過PC-2產生48位元的子金鑰

```python
def encrypt(plaintext, key):

    # Key generation
    # 1. Permuted choice 1: 64 to 56 bits
    key = permute(key, pc1)

    # 2. Spilting key
    left_key = key[0:28]
    right_key = key[28:56]

    # 3. Subkeys for 16 rounds
    subkeys = []
    for i in range(0, 16):
        # Shift
        left_key = shift_left(left_key, shift_table[i])
        right_key = shift_left(right_key, shift_table[i])

        # Permuted choice 2: 56 to 48 bits
        shifted_key = left_key + right_key
        shifted_key = permute(shifted_key, pc2)
        subkeys.append(shifted_key)
```

- **Encryption**



首尾各有一次置換、中間有16個相同的處理過程(round):

一、資料塊先被分成兩個32位元的半塊

二、「F函式」將資料半塊擴張，和當回次對應的子金鑰混合，之後進行置換。

三、F函式的輸出再與另一個半塊互斥或(XOR)

四、交換順序，進入下一個回次的處理。

```python
# Encryption
# Initial permutation
plaintext = permute(plaintext, initial_perm)

# Spilting plaintext
left_pt = plaintext[0:32]
right_pt = plaintext[32:64]

for i in range(0, 16):

    # The Feistel (F)
    # 1. Expansion: 32 to 48 bits
    right_expanded = permute(right_pt, expansion)

    # 2. Key mixing
    xor_right = xor(right_expanded, subkeys[i])

    # 3. Substitution
    substitution = ""
    for j in range(0, 8):
        row = int((xor_right[j*6] + xor_right[j*6 + 5]), 2)
        col = int((xor_right[j*6 + 1] + xor_right[j*6 + 2] +
                   xor_right[j*6 + 3] + xor_right[j*6 + 4]), 2)
        substitution += dec_toBinary(sbox[j][row][col])


    # 4. Permutation
    substitution = permute(substitution, per)

    # XOR left and F's output
    result = xor(left_pt, substitution)
    left_pt = result

    # Swap left and right
    if(i != 15):
        tmp = left_pt
        left_pt = right_pt
        right_pt = tmp
```

```python
# Final permutation after 16 rounds
combine = left_pt + right_pt
ciphertext = permute(combine, final_perm)

return ciphertext
```