# CSE 350/550: Assignment 2
# Project 0: DES

## 1. ENCRYPTION

Following are the main components of the encryption algorithm:

a. key generation

The function key_gen takes a 64 bit key K (i.e. 64 characters long bit string) and outputs 16 48 bit keys in the form of an array. The first part consists of converting the 64 bit key to a 56 bit key by dropping the parity bits. This is done using the key_parity array. After this 16 rounds of circular left shifting follow, where in each round the 56 bit key obtained above is split into 2 28 bit chunks, and each chunk is left shifted 1 or 2 bits depending on the round no, i.e. 1 bit left shifting for rounds 1,2,9 and 16, and 2 bits for the rest. These 2 chunks are finally combined and we use the key_compression array to create a 48 bit key from the previously resulting 56 bit key.

b. F box

The function F_box takes in a key K of 48 bits and a C of length 32 bits as input and includes 4 steps - expansion P box, XORing with K, S boxes and the straight P box. First of all, we expand the 32 bit C to a 48 bit C using the expansion_box array. This is followed by its xoring with the 48 bit key. We have implemented our custom XOR function that helps us XOR 2 bit strings of the same size. This output is then passed through the 8 S-boxes, by splitting it into 8 respective 6 bit blocks, and determining the row and column values using its bits 1,6 and the bits 2,3,4,5 respectively.
The output blocks are concatenated back to get the output which is passed through the straight P box using the array straight_permutation to get a 32 bit output.

c. Initial permutation

The function initial_permutation takes in a 64 bit string as an input, permutes it using the initial_lookup array, and returns the 64 bit permutation.

d. inverse initial permutation

The function inverse_initial_permutation takes in a 64 bit string as an input, permutes it using the final_lookup array (which is essentially the inverse of the initial_lookup array), and returns the 64 bit permutation.

e. 32 bit swap

The function swap_32_bit takes in a 64 bit string as input, splits it into 2 32 bit chunks, swaps them and returns the 64 bit result.

f. DES round

The function DES_round takes in a 64 bit string C and the 48 bit round key generated by key_gen as input. It then splits C into 2 32 bit chunks namely LE_i and RE_i. It feeds RE_i and the key K to the F_box, and xors this output with LE_i to get the new RE, i.e. RE_i_1 and assigns the initial RE i.e. RE_i to the new LE i,e, LE_i_1. It combines LE_i_1 and RE_i_1 to return the 64 bit output.

g. DES_encryption

The function DES_encryption takes in a 64 bit master key K as input, and a 64 bit plaintext P as input. It calls the key_gen function on K, to get the 16 round keys. Then it calls the initial_permutation function on P, which is followed by 16 calls to the DES_round function, which takes in the last intermediate ciphertext output and the corresponding round key. After this,the swap_32_bit function is called on the last intermediate cipher output of round 16 and the resultant is further fed into the inverse_initial_permutation function. All these intermediate ciphers along with the final ciphertext are stored in order in the intermediate_ciphers array and returned.

Note : all the permutation boxes and S boxes have been taken from here.

## 2. DECRYPTION

The function DES_decryption takes in a 64 bit master key K as input, and a 64 bit plaintext P as input. It calls the key_gen function on K, to get the 16 round keys. Then it reverses the order of the keys. Then it calls the initial_permutation function on P, which is followed by 16 calls to the DES_round function, which takes in the last intermediate ciphertext output and the corresponding round key. After this,the swap_32_bit function is called on the last intermediate cipher output of round 16 and the resultant is further fed into the inverse_initialintermediate_permutation function. All these intermediate ciphers along with the final ciphertext are stored in order in the intermediate_ciphers array and returned.

## 3. EXAMPLES

*Example1*

Key =

0000111100010101011100011100100101000111110110011111010000101100 1

```
Plaintext:  0x2468ace 0xeca86420
After encryption
C after initial_permutation:  0x5a005a00 0x3cf03c0f
C after round  1 :  0x3cf03c0f 0xbad22845
C after round  2 :  0xbad22845 0x99e9b723
C after round  3 :  0x99e9b723 0xbae3b9e
C after round  4 :  0xbae3b9e 0x42415649
C after round  5 :  0x42415649 0x18b3fa41
C after round  6 :  0x18b3fa41 0x9616fe23
C after round  7 :  0x9616fe23 0x67117cf2
C after round  8 :  0x67117cf2 0xc11bfc09
C after round  9 :  0xc11bfc09 0x887fbc6c
C after round  10 :  0x887fbc6c 0x600f7e8b
C after round  11 :  0x600f7e8b 0xf596506e
C after round  12 :  0xf596506e 0x738538b8
C after round  13 :  0x738538b8 0xc6a62c4e
C after round  14 :  0xc6a62c4e 0x56b0bd75
C after round  15 :  0x56b0bd75 0x75e8fd8f
C after round  16 :  0x75e8fd8f 0x25896490
C after 32 bit swap:  0x25896490 0x75e8fd8f
C after inverse_initial_permutation:  0xda02ce3a 0x89ecac3b
final C:  0xda02ce3a 0x89ecac3b
----------
After decryption
P2 after initial_permutation:  0x25896490 0x75e8fd8f
P2 after round  1 :  0x75e8fd8f 0x56b0bd75
P2 after round  2 :  0x56b0bd75 0xc6a62c4e
P2 after round  3 :  0xc6a62c4e 0x738538b8
P2 after round  4 :  0x738538b8 0xf596506e
P2 after round  5 :  0xf596506e 0x600f7e8b
P2 after round  6 :  0x600f7e8b 0x887fbc6c
P2 after round  7 :  0x887fbc6c 0xc11bfc09
P2 after round  8 :  0xc11bfc09 0x67117cf2
P2 after round  9 :  0x67117cf2 0x9616fe23
P2 after round  10 :  0x9616fe23 0x18b3fa41
P2 after round  11 :  0x18b3fa41 0x42415649
P2 after round  12 :  0x42415649 0xbae3b9e
P2 after round  13 :  0xbae3b9e 0x99e9b723
P2 after round  14 :  0x99e9b723 0xbad22845
P2 after round  15 :  0xbad22845 0x3cf03c0f
P2 after round  16 :  0x3cf03c0f 0x5a005a00
P2 after 32 bit swap:  0x5a005a00 0x3cf03c0f
P2 after inverse_initial_permutation:  0x2468ace 0xeca86420
```

A. Original plaintext = 0x2468ace 0xeca86420
   Ciphertext after decryption = 0x2468ace 0xeca86420
B. Output of 1st encryption round = 0x3cf03c0f 0xbad22845
   Output of 15th decryption round = 0xbad22845 0x3cf03c0f
C. Output of 14th encryption round = 0xc6a62c4e 0x56b0bd75
   Output of 2nd decryption round =0x56b0bd75 0xc6a62c4e

*Example2*

Key =
0111001000110110100010101101011110110001111000011011000010010001

```
Plaintext:  0x8172abe1 0x1b1c1c12
After encryption
C after initial_permutation:  0xaf2601d 0xd0e7496
C after round  1 :  0xd0e7496 0xe2de1b9
C after round  2 :  0xe2de1b9 0xb78e1533
C after round  3 :  0xb78e1533 0xe36d86ab
C after round  4 :  0xe36d86ab 0x4e77ba7f
C after round  5 :  0x4e77ba7f 0x768297f1
C after round  6 :  0x768297f1 0xdb05c8be
C after round  7 :  0xdb05c8be 0xf7064566
C after round  8 :  0xf7064566 0x4a58c712
C after round  9 :  0x4a58c712 0xf4ed7f1
C after round  10 :  0xf4ed7f1 0x7cc5e2b2
C after round  11 :  0x7cc5e2b2 0x57b918c1
C after round  12 :  0x57b918c1 0xc9455767
C after round  13 :  0xc9455767 0xb1d7ae46
C after round  14 :  0xb1d7ae46 0xda742580
C after round  15 :  0xda742580 0x6fe5d0de
C after round  16 :  0x6fe5d0de 0x3f0cd194
C after 32 bit swap:  0x3f0cd194 0x6fe5d0de
C after inverse_initial_permutation:  0xe4c2f3d2 0x4fe0ae2f
final C:  0xe4c2f3d2 0x4fe0ae2f

After decryption
P2 after initial_permutation:  0x3f0cd194 0x6fe5d0de
P2 after round  1 :  0x6fe5d0de 0xda742580
P2 after round  2 :  0xda742580 0xb1d7ae46
P2 after round  3 :  0xb1d7ae46 0xc9455767
P2 after round  4 :  0xc9455767 0x57b918c1
P2 after round  5 :  0x57b918c1 0x7cc5e2b2
P2 after round  6 :  0x7cc5e2b2 0xf4ed7f1
P2 after round  7 :  0xf4ed7f1 0x4a58c712
P2 after round  8 :  0x4a58c712 0xf7064566
P2 after round  9 :  0xf7064566 0xdb05c8be
P2 after round  10 :  0xdb05c8be 0x768297f1
P2 after round  11 :  0x768297f1 0x4e77ba7f
P2 after round  12 :  0x4e77ba7f 0xe36d86ab
P2 after round  13 :  0xe36d86ab 0xb78e1533
P2 after round  14 :  0xb78e1533 0xe2de1b9
P2 after round  15 :  0xe2de1b9 0xd0e7496
P2 after round  16 :  0xd0e7496 0xaf2601d
P2 after 32 bit swap:  0xaf2601d 0xd0e7496
P2 after inverse_initial_permutation:  0x8172abe1 0x1b1c1c12
```

A. Original plaintext = 0x8172abe1 0x1b1c1c12
   Ciphertext after decryption = 0x8172abe1 0x1b1c1c12
B. Output of 1st encryption round = 0xd0e7496 0xe2de1b9
   Output of 15th decryption round = 0xe2de1b9 0xd0e7496
C. Output of 14th encryption round = 0xb1d7ae46 0xda742580
   Output of 2nd decryption round = 0xda742580 0xb1d7ae46