# CryptoGraphy DA2

**Question:-**
**Encode a message using DES when encrypted. Try the following experiments and note how they change the output:**

**a) Change one character at the end of the message. How much of the encoded message changes?**
**b) Change one character at the beginning of the message. How much of the encoded message changes?**
**c) Delete one character at the end of the message. How much of the encoded message changes?**
**d) Change one character in the key. How much of the encoded message changes?**
**e) Decrypt a message using a key with one character changed. Does it look anything like the original?**

# Solution:-

## Code:-

```
--------------------------------

k1 = 0xAABB09182736CCDD

k2 = 0xAABB09182736CCDE  # changed one character at the end of the key.

#k3 = 0x133457799BBCDFF1

m1 = 0x123456ABCD132536

#m2 = 0x0123456789ABCDEF

m3 = 0x123456ABCD132537  # changed one character at the end of the message.

m4 = 0x223456ABCD132536  # changed one character at the beginning of the message

m5 = 0x123456ABCD13253  # Delete one character at the end of the message

key1 = "AABB09182736CCDD"
```

```
key2 = "133457799BBCDFF1"


PC1 = (
        57, 49, 41, 33, 25, 17, 9,
        1,  58, 50, 42, 34, 26, 18,
        10, 2,  59, 51, 43, 35, 27,
        19, 11, 3,  60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7,  62, 54, 46, 38, 30, 22,
        14, 6,  61, 53, 45, 37, 29,
        21, 13, 5,  28, 20, 12, 4
)

PC2 = (
        14, 17, 11, 24, 1, 5,
        3, 28, 15, 6, 21, 10,
        23, 19, 12, 4, 26, 8,
        16, 7, 27, 20, 13, 2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32
)
Sboxes = {
        0: (
        14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
        0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
        4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
        15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13
        ),
        1: (
        15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
        3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
        0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
        13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9
        ),
        2: (
        10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
        13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
        13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
        1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12
        ),
        3: (
        7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
        13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
        10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
        3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14
        ),
        4: (
        2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
        14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
        4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
        11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3
        ),
        5: (
        12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
        10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
        9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
        4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13
        ),
```

```
        6: (
        4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
        13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
        1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
        6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12
        ),
        7: (
        13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
        1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
        7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
        2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11
        )
}
def key_generation(key):
        key_bin = "{0:08b}".format(int(key, 16))
        y=64-len(key_bin)
        key_bin=str(y*str(0))+str(key_bin)
        bit56_key=''
        for x in PC1:
        bit56_key=bit56_key+key_bin[x-1]
        l=bit56_key[:28]
        r=bit56_key[28:]
        l11=[]
        for i in l:
        l11.append(i)
        def circularshift1(left_key):
        l1=[]
        for i in left_key:
        l1.append(i)

        y=l1[0]
        l1.remove(l1[0])
        l1.append(y)
        return l1

        shifted_keys=[]
        shifted_keys.append(circularshift1(l));shifted_keys.append(circularshift1(l))

        shifted_keys2=[]
        for i in range(0,170):
        shifted_keys.append(circularshift1(shifted_keys[i]))

        for i in range(3,100,4):
        if i==31:
        break
        shifted_keys2.append(shifted_keys[i])

        shifted_keys2.append(circularshift1(shifted_keys2[6]))
        l12=circularshift1(l11)
        shifted_keys2.insert(0,l12)

        shifted_keys3=[]
        shifted_keys4=[]
        shifted_keys_original_left=[]

        shifted_keys3.append(circularshift1(circularshift1(shifted_keys2[-1])))


        for i in range(0,100):
        d = shifted_keys3[-1]
        shifted_keys3.append(circularshift1(d))
        for i in range(0,13,2):
```

```
shifted_keys4.append(shifted_keys3[i])
shifted_keys4.append(circularshift1(shifted_keys4[-1]))

for i in range(len(shifted_keys2)):
shifted_keys_original_left.append(shifted_keys2[i])
for i in range(len(shifted_keys4)):
shifted_keys_original_left.append(shifted_keys4[i])

r11=[]
for i in r:
r11.append(i)
def circularshift2(right_key):
r1=[]
for i in right_key:
r1.append(i)
y=r1[0]
r1.remove(r1[0])
r1.append(y)
return r1

shifted_keys_r=[]
shifted_keys_r.append(circularshift2(r));shifted_keys_r.append(circularshift2(r))

shifted_keys2_r=[]
for i in range(0,170):
shifted_keys_r.append(circularshift2(shifted_keys_r[i]))

for i in range(3,100,4):
if i==31:
break
shifted_keys2_r.append(shifted_keys_r[i])

shifted_keys2_r.append(circularshift2(shifted_keys2_r[6]))
r12=circularshift2(r11)
shifted_keys2_r.insert(0, r12)

shifted_keys3_r=[]
shifted_keys4_r=[]
shifted_keys_original_right=[]

shifted_keys3_r.append(circularshift2(circularshift2(shifted_keys2_r[-1])))

for i in range(0,100):
d = shifted_keys3_r[-1]
shifted_keys3_r.append(circularshift2(d))
for i in range(0,13,2):
shifted_keys4_r.append(shifted_keys3_r[i])
shifted_keys4_r.append(circularshift2(shifted_keys4_r[-1]))

for i in range(len(shifted_keys2_r)):
shifted_keys_original_right.append(shifted_keys2_r[i])

for i in range(len(shifted_keys4_r)):
shifted_keys_original_right.append(shifted_keys4_r[i])


shifted_keys_original=[]
k = 0
for i in shifted_keys_original_right:
for j in i:
shifted_keys_original_left[k].append(j)
k = k+1
```

```
            shifted_keys_original = shifted_keys_original_left
            final_key=[]
            q=0
            for i in shifted_keys_original:
            for j in PC2:
            final_key.append(shifted_keys_original[q][j-1])
            q=q+1
            m=0
            final_key_original=[]

            for i in range(0,len(final_key),48):
            final_key_original.append(final_key[m:i])
            m=i

            final_key_original.remove([])
            print('Keys:-')
            for i in final_key_original:
            for j in i:
            print(j,end='')
            print()

IP = (
            58, 50, 42, 34, 26, 18, 10, 2,
            60, 52, 44, 36, 28, 20, 12, 4,
            62, 54, 46, 38, 30, 22, 14, 6,
            64, 56, 48, 40, 32, 24, 16, 8,
            57, 49, 41, 33, 25, 17, 9, 1,
            59, 51, 43, 35, 27, 19, 11, 3,
            61, 53, 45, 37, 29, 21, 13, 5,
            63, 55, 47, 39, 31, 23, 15, 7
)
IP_INV = (
            40, 8, 48, 16, 56, 24, 64, 32,
            39, 7, 47, 15, 55, 23, 63, 31,
            38, 6, 46, 14, 54, 22, 62, 30,
            37, 5, 45, 13, 53, 21, 61, 29,
            36, 4, 44, 12, 52, 20, 60, 28,
            35, 3, 43, 11, 51, 19, 59, 27,
            34, 2, 42, 10, 50, 18, 58, 26,
            33, 1, 41, 9, 49, 17, 57, 25
)

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



P = (
            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
)

def encrypt(msg, key, decrypt=False):
        assert isinstance(msg, int) and isinstance(key, int)
        assert not msg.bit_length() > 64
        assert not key.bit_length() > 64
        key = per_by_table(key, 64, PC1)
        C0 = key >> 28
        D0 = key & (2 ** 28 - 1)
        round_keys = gen_ys(C0, D0)

        msg_block = per_by_table(msg, 64, IP)
        L0 = msg_block >> 32
        R0 = msg_block & (2 ** 32 - 1)

        L_last = L0
        R_last = R0
        for i in range(1, 17):
        if decrypt:
        i = 17 - i
        L_round = R_last
        R_round = L_last ^ r_fun(R_last, round_keys[i])
        L_last = L_round
        R_last = R_round
        cipher_block = (R_round << 32) + L_round
        cipher_block = per_by_table(cipher_block, 64, IP_INV)

        return cipher_block


def r_fun(Ri, Ki):
        Ri = per_by_table(Ri, 32, E)

        Ri ^= Ki
        Ri_blocks = [((Ri & (0b111111 << shift_val)) >> shift_val) for shift_val in (42, 36, 30, 24, 18, 12, 6, 0)]
        for i, block in enumerate(Ri_blocks):
        row = ((0b100000 & block) >> 4) + (0b1 & block)
        col = (0b011110 & block) >> 1
        Ri_blocks[i] = Sboxes[i][16 * row + col]

        Ri_blocks = zip(Ri_blocks, (28, 24, 20, 16, 12, 8, 4, 0))
        Ri = 0
        for block, lshift_val in Ri_blocks:
        Ri += (block << lshift_val)
        Ri = per_by_table(Ri, 32, P)

        return Ri


def per_by_table(block, block_len, table):
        block_str = bin(block)[2:].zfill(block_len)
        perm = []
        for pos in range(len(table)):
        perm.append(block_str[table[pos] - 1])
        return int(''.join(perm), 2)


def gen_ys(C0, D0):
        round_keys = dict.fromkeys(range(0, 17))
        lrot_values = (1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1)
```

```
        lrot = lambda val, r_bits, max_bits: \
        (val << r_bits % max_bits) & (2 ** max_bits - 1) | \
        ((val & (2 ** max_bits - 1)) >> (max_bits - (r_bits % max_bits)))
        C0 = lrot(C0, 0, 28)
        D0 = lrot(D0, 0, 28)
        round_keys[0] = (C0, D0)
        for i, rot_val in enumerate(lrot_values):
        i += 1
        Ci = lrot(round_keys[i - 1][0], rot_val, 28)
        Di = lrot(round_keys[i - 1][1], rot_val, 28)
        round_keys[i] = (Ci, Di)
        del round_keys[0]
        for i, (Ci, Di) in round_keys.items():
        Ki = (Ci << 28) + Di
        round_keys[i] = per_by_table(Ki, 56, PC2)
        return round_keys



def Final(key, msg):
        print('KEY:          {:x}'.format(key))
        print('MESSAGE:   {:x}'.format(msg))
        cipher_text = encrypt(msg, key)
        print('ENCRYPTED TEXT: {:x}'.format(cipher_text))
        plain_text = encrypt(cipher_text, key, decrypt=True)
        print('DECRYPTED TEXT: {:x}'.format(plain_text))



#key_generation(key1)
print('Original')
Final(k1, m1)
print('----------------------')
print('Change one character at the end of the message')
#key_generation(key2)
Final(k1, m3)
print('----------------------')
print('Change one character at the beginning of the message')
Final(k1, m4)
print('----------------------')
print('Delete one character at the end of the message.')
Final(k1, m5)
print('----------------------')
print('Change one character in the key and  Decrypt a message using a key with one character changed ')
Final(k2, m1)
```

---

# Input:- None

# Output:-

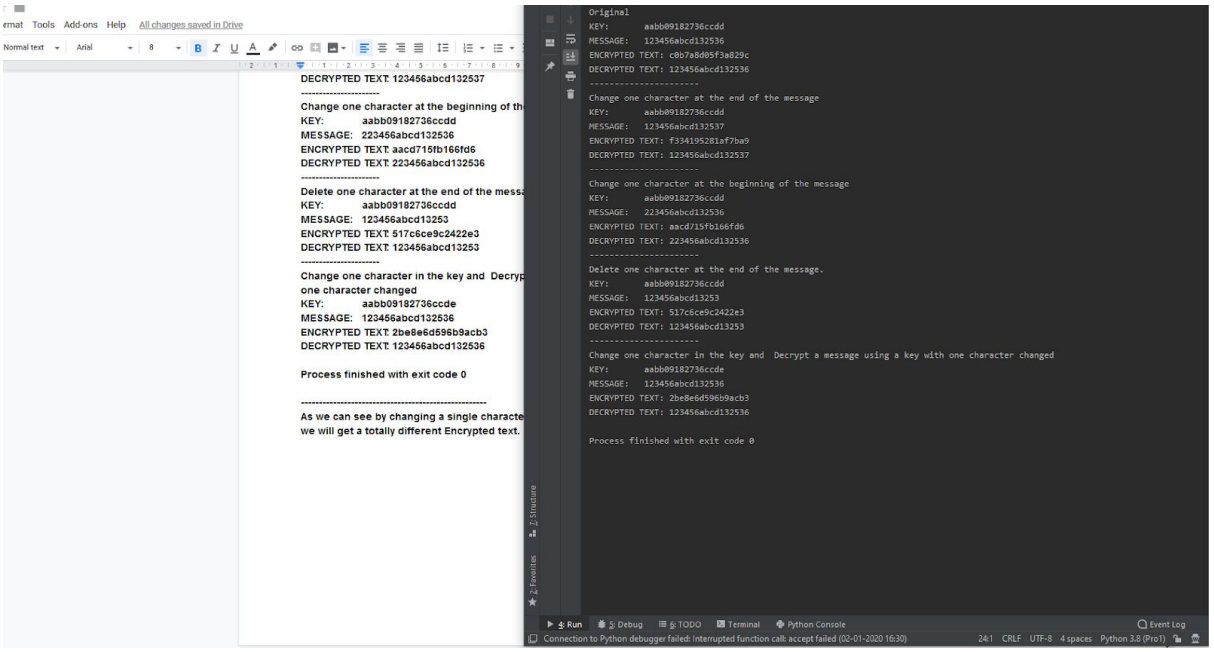
**Original**
**KEY:  aabb09182736ccdd**
**MESSAGE:  123456abcd132536**

**ENCRYPTED TEXT: c0b7a8d05f3a829c**
**DECRYPTED TEXT: 123456abcd132536**
**----------**
**Change one character at the end of the message**
**KEY: aabb09182736ccdd**
**MESSAGE: 123456abcd132537**
**ENCRYPTED TEXT: f334195281af7ba9**
**DECRYPTED TEXT: 123456abcd132537**
**----------**
**Change one character at the beginning of the message**
**KEY: aabb09182736ccdd**
**MESSAGE: 223456abcd132536**
**ENCRYPTED TEXT: aacd715fb166fd6**
**DECRYPTED TEXT: 223456abcd132536**
**----------**
**Delete one character at the end of the message.**
**KEY: aabb09182736ccdd**
**MESSAGE: 123456abcd13253**
**ENCRYPTED TEXT: 517c6ce9c2422e3**
**DECRYPTED TEXT: 123456abcd13253**
**----------**
**Change one character in the key and Decrypt a message using a key with one character changed**
**KEY: aabb09182736ccde**
**MESSAGE: 123456abcd132536**
**ENCRYPTED TEXT: 2be8e6d596b9acb3**
**DECRYPTED TEXT: 123456abcd132536**

**Process finished with exit code 0**

**----------**
**As we can see by changing a single character in the key or in the message we will get a totally different Encrypted text.**

**I've attached a screenshot of the output screen.**




# ID -**********

# Name- Hritish kumar