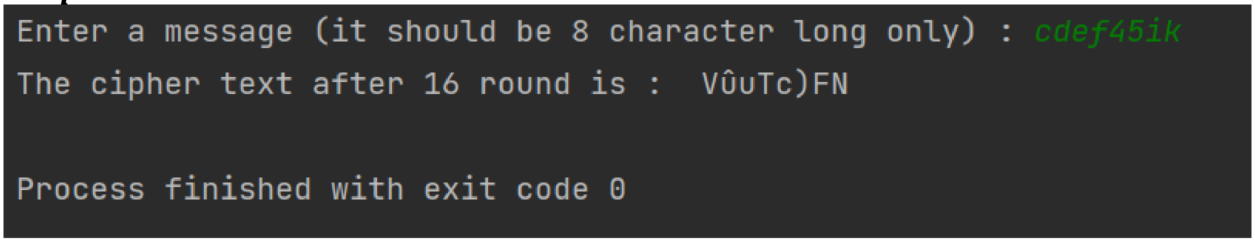
Source Code :

```
# DES
import random

def DESWorking(binary_input: str):

    left = binary_input[:32]
    right = binary_input[32:]
        # print(left + right)
    k = random.randint(1, 42949672)
    k = format(k, '032b')
    xor = []
    for i in range(32):
        xor.append(int(right[i]) ^ int(k[i]))
    new_xor = []
    for i in range(32):
        new_xor.append(int(xor[i]) ^ int(left[i]))
    right = "".join(str(i) for i in new_xor)
    return right, left



if __name__ == "__main__":
    message = input("Enter a message (it should be 8 character long only) : ")
    binary_of_message = "".join(format(ord(i), '08b') for i in message)
    k = 0
    left = 0
    right = 0
    while k < 16:
        left, right = DESWorking(binary_of_message)
        binary_of_message = left + right
        k += 1
    total = left + right
    print("The cipher text after 16 round is : ", end=" ")
    for i in range(0, len(total), 8):
        print(chr(int(total[i:i+8], 2)), end="")
    print()
```

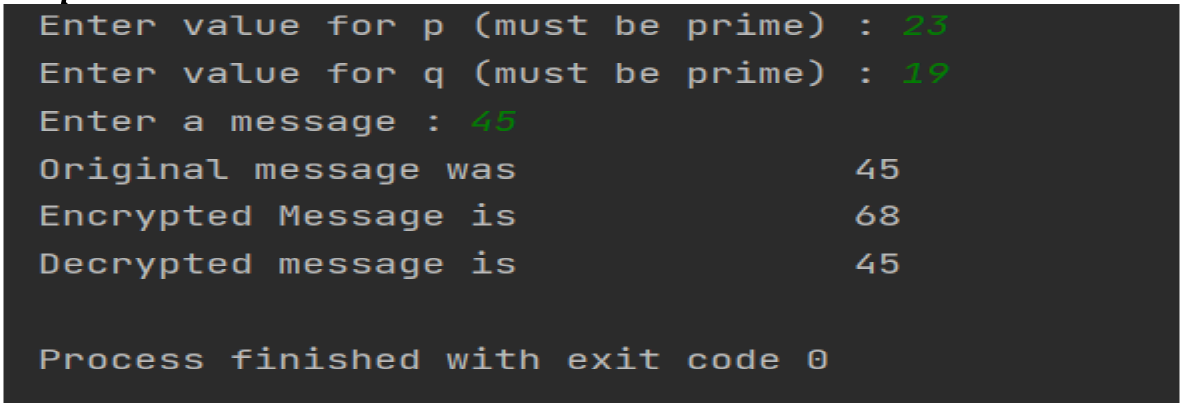
Output:



```
Enter a message (it should be 8 character long only) : cdef45ik
The cipher text after 16 round is :  VûuTc)FN


Process finished with exit code 0
```

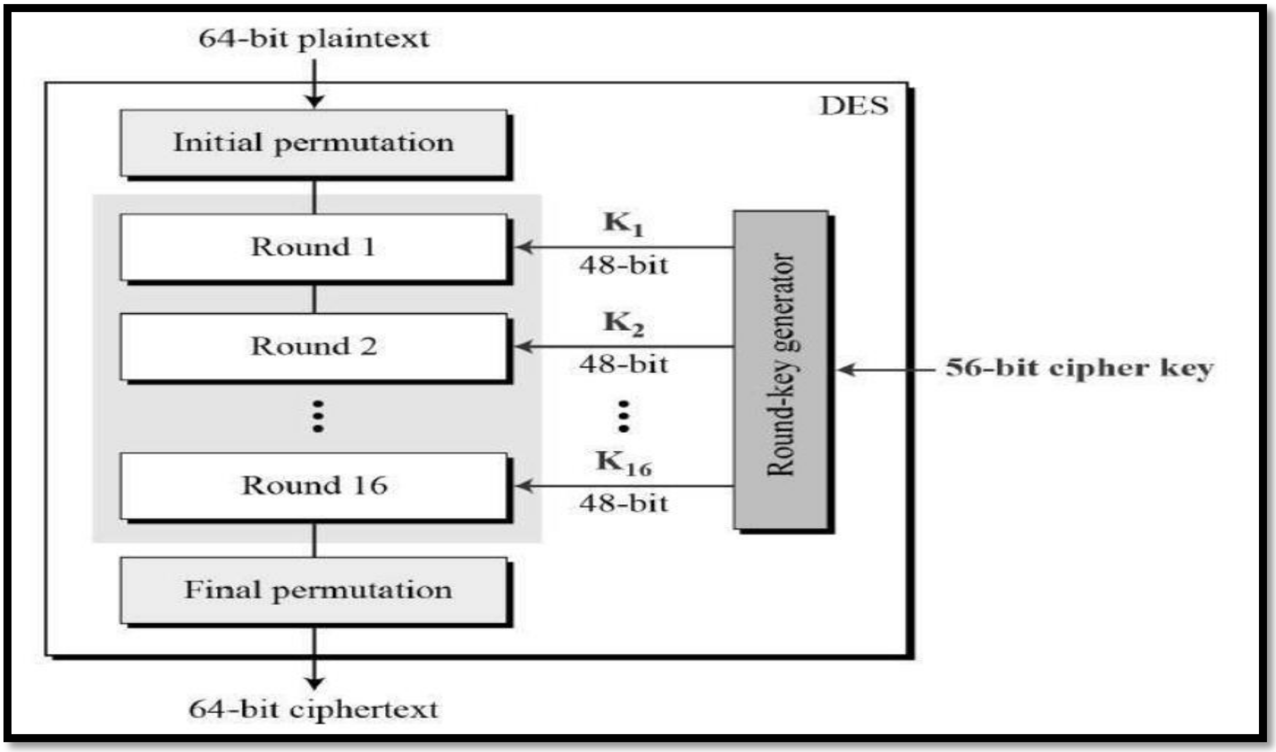
```
# RSA

import random

def gcdByEuclideanMethod(a, b):
    return a if b == 0 else gcdByEuclideanMethod(b, a%b)

def encryptMessage(message, e, n):
   cipher = (message ** e) % n
   return cipher

def decryptMessage(cipher, d, n):
   plain = (cipher ** d) % n
   return plain

if __name__ == '__main__':
   p = int(input("Enter value for p (must be prime) : "))
   q = int(input("Enter value for p (must be prime) : "))
   n = p * q
   phi_n = (p-1) * (q-1)
   e = [i for i in range(3, phi_n, 2) if gcdByEuclideanMethod(phi_n, i) == 1]
   e = e[random.randint(0, len(e))]
   d = [i for i in range(3, phi_n) if (i * e) % phi_n == 1]
   d = d[random.randint(0, len(d)-1)]
   # print(e, d)
   message = int(input("Enter a message : "))
   cipher = encryptMessage(message, e, n)
   decrypted = decryptMessage(cipher, d, n)
   # print("Public key <{}, {}> \nPrivate key <{}, {}>".format(e, n, d, n))
   print("Original message was {:15}\nEncrypted Message is {:15}\nDecrypted
message is {:15}".format(message, cipher, decrypted))
```

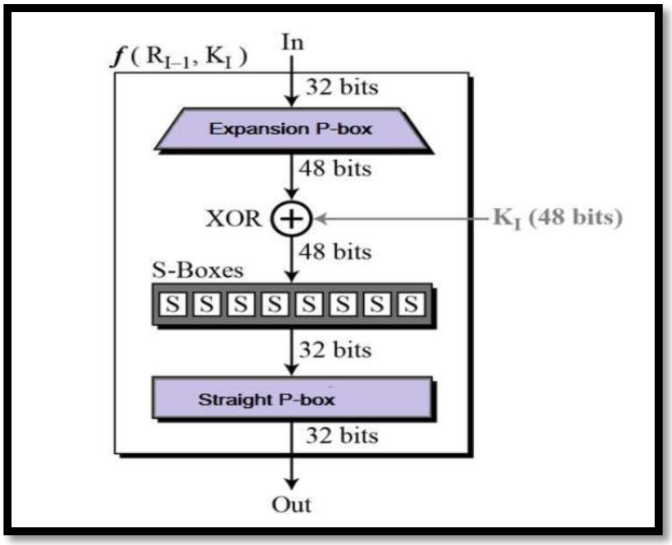
Output :

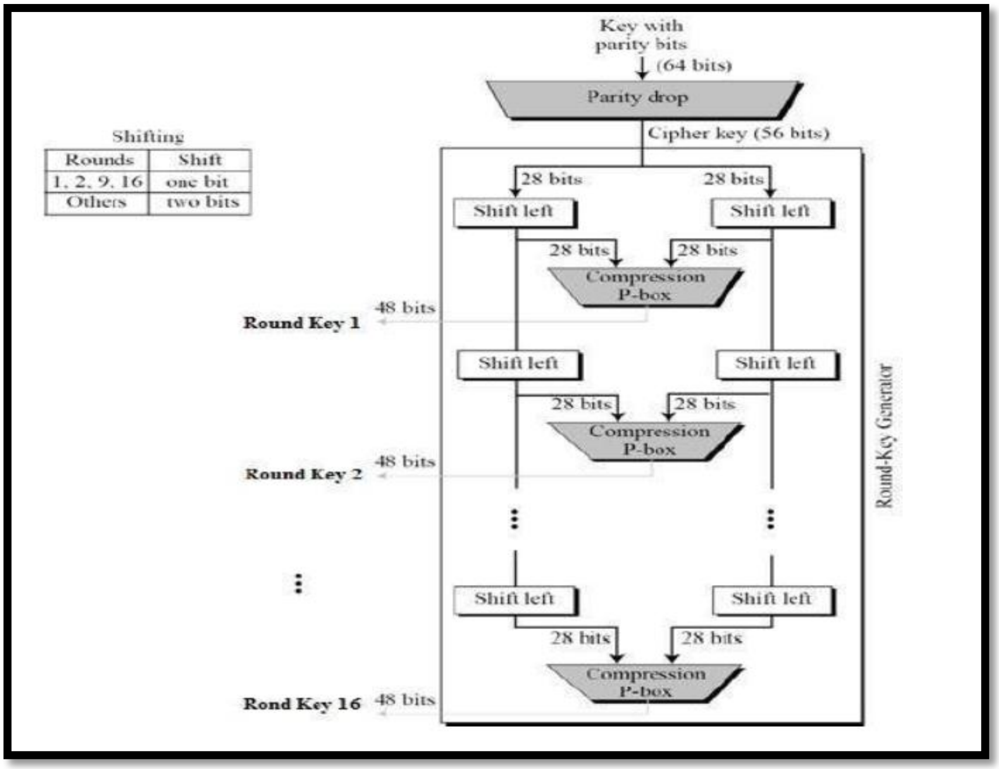


```
Enter value for p (must be prime) : 23
Enter value for q (must be prime) : 19
Enter a message : 45
Original message was              45
Encrypted Message is              68
Decrypted message is              45

Process finished with exit code 0
```

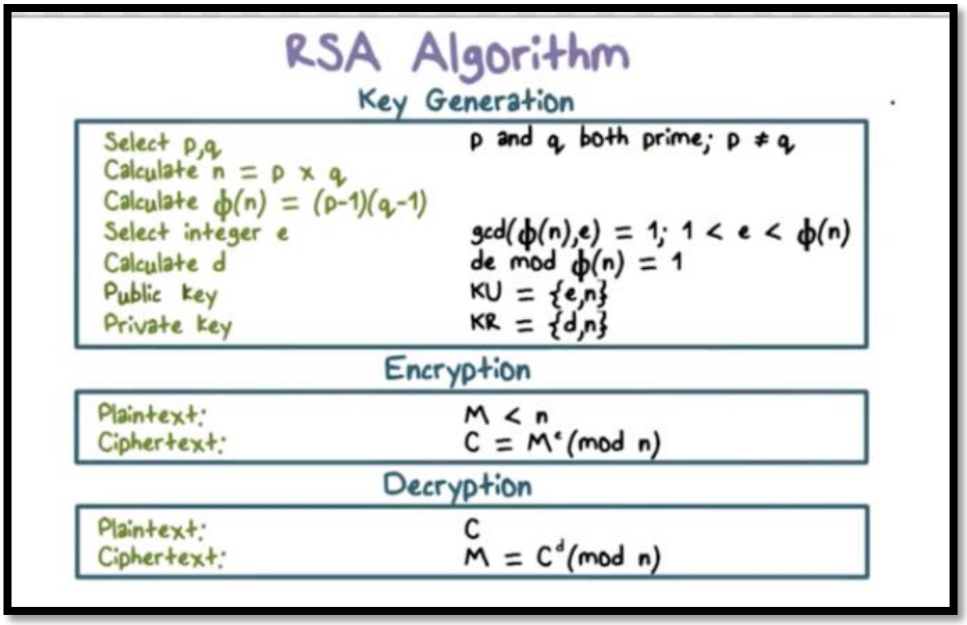
DES:


64-bit plaintext
DES
Initial permutation
Round 1
Round 2
Round 16
Final permutation
K_1
48-bit
K_2
48-bit
K_{16}
48-bit
Round-key generator
56-bit cipher key
64-bit ciphertext


$f(R_{I-1}, K_I)$
In
32 bits
Expansion P-box
48 bits
XOR
K_I (48 bits)
48 bits
S-Boxes
S S S S S S S S
32 bits
Straight P-box
32 bits
Out



Key generation in DES



RSA Algorithm

Key Generation

Select p,q	p and q both prime; $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n),e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e,n\}$
Private key	$KR = \{d,n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e (\text{mod } n)$

Decryption

Plaintext:	C
Ciphertext:	$M = C^d (\text{mod } n)$