```
import hashlib
import base64

ROUNDS = 8
BLOCKSIZE = 8
SECRET = "3f788083-77d3-4502-9d71-21319f1792b6"

def encryptMessage(key, message, mode):
    ciphertext = ""
    n = BLOCKSIZE
    message = [message[i: i + n] for i in range(0, len(message), n)]
    lengthOfLastBlock = len(message[-1])

    if lengthOfLastBlock < BLOCKSIZE:
        message[-1] += " " * (BLOCKSIZE - lengthOfLastBlock)

    key = key_256(key)
    key_initial = key

    for block in message:
        L = ["" for _ in range(ROUNDS + 1)]
        R = ["" for _ in range(ROUNDS + 1)]
        L[0] = block[:BLOCKSIZE // 2]
        R[0] = block[BLOCKSIZE // 2:]

        for i in range(1, ROUNDS + 1):
            L[i] = R[i - 1]

            if mode == "cbc":
                key = key_initial if i == 1 else subkeygen(L[i], key_initial,
i)

            R[i] = xor(L[i - 1], scramble(R[i - 1], i, key))
            ciphertext += (L[ROUNDS] + R[ROUNDS])

    return ciphertext

def decryptCipher(key, ciphertext, mode):
    message = ""
    n = BLOCKSIZE
    ciphertext = [ciphertext[i: i + n] for i in range(0, len(ciphertext), n)]
    lengthOfLastBlock = len(ciphertext[-1])

    if lengthOfLastBlock < BLOCKSIZE:
        ciphertext[-1] += " " * (BLOCKSIZE - lengthOfLastBlock)

    key = key_256(key)
```

```
    key_initial = key

    for block in ciphertext:
        L = ["" for _ in range(ROUNDS + 1)]
        R = ["" for _ in range(ROUNDS + 1)]
        L[ROUNDS] = block[:BLOCKSIZE // 2]
        R[ROUNDS] = block[BLOCKSIZE // 2:]

        for i in range(8, 0, -1):
            if mode == "cbc":
                key = subkeygen(L[i], key_initial, i)

            if i == 1:
                key = key_initial

            R[i - 1] = L[i]
            L[i - 1] = xor(R[i], scramble(L[i], i, key))
            message += (L[0] + R[0])

    return message

def key_256(key):
    return hashlib.sha256((key + SECRET).encode()).hexdigest()

def subkeygen(s1, s2, i):
    return hashlib.sha256((s1 + s2).encode()).hexdigest()

def scramble(x, i, k):
    k = stobin(k)
    x = stobin(str(x))
    k = bintoint(k)
    x = bintoint(x)
    res = pow((x * k), i)
    res = itobin(res)
    return bintostr(res)

def xor(s1, s2):
    return ''.join(chr(ord(a) ^ ord(b)) for a, b in zip(s1, s2))

def stobin(s):
    return ''.join('{:08b}'.format(ord(c)) for c in s)

def bintoint(s):
    return int(s, 2)

def itobin(i):
    return bin(i)
```

```
def bintostr(b):
    n = int(b, 2)
    return ''.join(chr(int(b[i: i + 8], 2)) for i in range(0, len(b), 8))

mode = "ecb"
key = "qwerty"
input_text = "transfer one million dollars"

print("Input text:", input_text)
print("Key:", key)
print("Mode:", mode)
print("\n==================")

cipher = encryptMessage(key, input_text, mode)
print("Cipher:", base64.b64encode(cipher.encode()).decode())

plain = decryptCipher(key, cipher, mode)
print("Plain text:", plain)
```

output :-

```
Input text: transfer one million dollars

Key: qwerty
Mode: ecb

==================
Cipher: aDQKQWHDqcKqw600KcKXKS4vY2Vtw6lKOD0WwpVWegXCv8OXKxjCg8OB
Plain text: transfer one million dollars
```