### DES算法程序说明

1. 定义眀密文类

class Text():

    def \_\_init\_\_(self,array=None) -> None:

        self.n=8

        if array is None:

            self.array = np.array([i for i in range(self.n)]#眀密文索引从0开始

                                , dtype=np.int8)

        else:

            self.array=array

class Plaintext(Text):

    def \_\_init\_\_(self,array) -> None:

        super().\_\_init\_\_(array)

        self.P = self.array

class Ciphertext(Text):

    def \_\_init\_\_(self,array) -> None:

        super().\_\_init\_\_(array)

        self.C = self.array

明文类Plaintext与密文类Ciphertext均继承自Text类。Text对眀密文的分组长度作出了设定。

1. 定义初始IP与最终置换

def IP(text:Text)->Text:

    old=text

    index=(2,6,3,1,4,8,5,7)

    index=tuple(value-1 for value in index)

    new=Text()

    for i,j in enumerate(index):

        new.array[i]=old.array[j]

    return new

def Inverse\_IP(text:Text)->Text:

    old=text

    index=(4,1,3,5,7,2,8,6)

    index=tuple(value-1 for value in index)

    new=Text()

    for i,j in enumerate(index):

        new.array[i]=old.array[j]

    return new

1. 定义密钥类

#密钥

class Key():

    def \_\_init\_\_(self,length:int,key=None) -> None:

        self.length=length

        if key is None:

            self.key = np.array([i for i in range(length)],#密钥索引从0开始

                           dtype=np.int8)

        else:

            self.key=key

类Key可以自由设定密钥的长度，方便后续子密钥的生成。

1. 定义子密钥生成函数

def P\_10(key:Key)->Key:

    old=key

    index=(3,5,2,7,4,10,1,9,8,6)

    index=tuple(value-1 for value in index)

    new=Key(length=10)

    for i,j in enumerate(index):

        new.key[i]=old.key[j]

return new

def P\_8(key:Key)->Key:

    old=key

    index=(6,3,7,4,8,5,10,9)

    index=tuple(value-1 for value in index)

    new=Key(length=8)

    for i in range(8):

        new.key[i]=old.key[index[i]]

return new

def left\_shift(key:Key,shift\_amount=1)->Key:

    d=shift\_amount #左移位数

    old=key

    new=Key(old.length)

    for i in range(5):#0-4

        new.key[(i-d)%5]=old.key[i]

        new.key[(i-d)%5+5]=old.key[i+5]

return new

def get\_k1(K:Key)->Key:

    after\_P10=P\_10(K)

    after\_shift=left\_shift(after\_P10,shift\_amount=1)

    after\_P8=P\_8(after\_shift)

    return after\_P8

def get\_k2(K:Key)->Key:

    after\_P10=P\_10(K)

    after\_shift=left\_shift(after\_P10,shift\_amount=2)

    after\_P8=P\_8(after\_shift)

    return after\_P8

1. S-DES 函数

class S\_DES\_fk():

    #key:Key 对应ki

    def \_\_init\_\_(self,key:Key,text:Text) -> None:

        self.EPBox=[4,1,2,3,2,3,4,1]

        self.SBox\_1=[[[0,1],[0,0],[1,1],[1,0]],

                     [[1,1],[1,0],[0,1],[0,0]],

                     [[0,0],[1,0],[0,1],[1,1]],

                     [[1,1],[0,1],[0,0],[1,0]]]

        self.SBox\_2=[[[0,0],[0,1],[1,0],[1,1]],

                     [[1,0],[1,1],[0,1],[0,0]],

                     [[1,1],[0,0],[0,1],[0,0]],

                     [[1,0],[0,1],[0,0],[1,1]]]

        self.SPBox=[2,4,3,1]

        self.key=key

        self.L\_text=text.array[0:len(text.array)//2].copy()

        self.R\_text=text.array[len(text.array)//2:len(text.array)].copy()

    def fun\_EPBox(self)->np.array:

        old=self.R\_text

        index=self.EPBox

        index=tuple(value-1 for value in index)

        new=Text()

        for i,j in enumerate(index):

            new.array[i]=old[j]

        return new.array

    def fun\_XOR(self)->np.array:

        array\_1=self.fun\_EPBox()

        array\_2=self.key.key

        result=array\_1^array\_2 # 使用^进行元素级异或操作

        return result

    def fun\_S\_Boxs(self)->np.array:

        array=self.fun\_XOR()

        i=array[0]\*2+array[3]

        j=array[1]\*2+array[2]

        res1=self.SBox\_1[i][j]

        i=array[4]\*2+array[7]

        j=array[5]\*2+array[6]

        res2=self.SBox\_2[i][j]

        result=res1+res2

        return result

    def fun\_SPBox(self)->np.array:

        old=self.fun\_S\_Boxs()

        index=self.SPBox

        index=tuple(value-1 for value in index)

        new=np.array([0,0,0,0])

        for i,j in enumerate(index):

            new[i]=old[j]

        return new

    #轮函数F输出结果

    def fun\_F(self)->np.array:

        return self.fun\_SPBox()

    #S-DES函数最终输出值

    def output(self)->Text:

        array\_F=self.fun\_F()

        res\_XOR=self.L\_text^array\_F

        output = np.concatenate((res\_XOR, self.R\_text))

        return Text(output)

    def outputAfterSWAP(self)->Text:

        array\_F=self.fun\_F()

        res\_XOR=self.L\_text^array\_F

        output = np.concatenate((self.R\_text, res\_XOR))

        return Text(output)

outputAfterSWAP()函数是将S-DES函数最终输出值进行左右互换SW操作