《信息安全基础》实验报告

一、实验目的

掌握频度分析法原理和 Feistel 加解密原理

二、实验项目内容

1. 使用频度分析法解密文本，并写出替换表

2. 编程实现 Feistel 加密/解密

三、实验设计

1. 频度分析法

频度分析法是一种经典的密码分析方法，用于破译基于替换密码的加密文本。该方法的基本思想是通过分析加密文本中字符的频率分布，推断出每个字符在加密前的明文字符。这是因为明文中某些字符出现的频率比其他字符更高，因此在加密后的文本中，使用相同的加密算法加密这些字符后，它们出现的频率仍然很高。对于任何一种书面语言而言，不同的字母或字母组合出现的频率各不相同。但统计表明，如果以该语言书写足够长的文本，都呈现出大致相同的特征字母分布规律如下：

English letter frequencies.

E	12.3%	R	6.0%	F	2.3%	K	0.5%
T	9.6%	H	5.1%	M	2.3%	Q	0.2%
A	8.1%	L	4.0%	W	2.0%	X	0.2%
O	7.9%	D	3.7%	Y	1.9%	J	0.1%
N	7.2%	C	3.2%	B	1.6%	Z	0.1%
I	7.2%	U	3.1%	G	1.6%		
S	6.6%	P	2.3%	V	0.9%		

频度分析法通常分为两个步骤：

（1）分析加密文本中字符的频率分布，确定每个密文字符最有可能对应哪个明文字符；

（2）重复（1）的步骤，结合自然语言规范判断破译的正确性；

（3）根据分析结果，构建密文字符到明文字符的替换表，从而实现对加密文本的解密。

在实际应用中，为了提高频率分析法的分析效率，通常会先破解出一部分明文，然后根据这部分明文推断出更多的密钥，最终得到完整的明文。频度分析法的破解效果取决于加密算法和密钥长度。对于使用单

一替换表的加密算法，密钥长度较短的情况下，频度分析法通常可以有效破解加密文本。但是，对于一些更加复杂的加密算法，如多重替换等，频度分析法的效果会受到影响。

2. Feistel 加密/解密

Feistel 网络是一种经典的对称加密算法结构，它由两个相同的部分组成，一个用于加密，另一个用于解密，每个部分包含若干轮计算。在加密时，明文会被分成 L 和 R 两部分，每个部分经过一定次数的轮函数的变换，然后将它们合并起来。解密时，密文也会被分成两部分，然后经过相同的轮函数的逆变换，最终合并得到明文。

在 Feistel 网络中，轮函数（即 F）的设计是至关重要的，这关系到结果的正确性。它必须满足两个条件：可逆性和混淆性，即轮函数必须能够通过一个相反的运算来撤销它的效果，并且轮函数必须能够将输入数据与加密密钥混淆起来，使得攻击者难以通过对密文的分析来推断出加密密钥。在本实验中轮函数采用的是 SHA256 哈希函数，函数接受中间结果 Ri 和密钥 Ki，返回一个固定长度（256 位）的二进制数。

四、实验过程或算法

1. 频度分析法

密文：UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFPF APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ

（1）统计密文中每个字母频率，按从高到低排序如下：

```
P: 13.333333%  Q: 2.500000%
Z: 11.666667%  T: 2.500000%
U: 8.333333%   G: 1.666667%
S: 8.333333%   B: 1.666667%
O: 7.500000%   A: 1.666667%
M: 6.666667%   Y: 1.666667%
H: 5.833333%   I: 0.833333%
E: 5.000000%   J: 0.833333%
D: 5.000000%   N: 0.000000%
V: 4.166667%   R: 0.000000%
X: 4.166667%   L: 0.000000%
W: 3.333333%   C: 0.000000%
F: 3.333333%   K: 0.000000%
```

（2）假设频率最高的字母 P 由 E 替换得到，将密文中 P 还原为 E，得到：UZ QSO VUOHXMOEV GPOZEEVSG ZWSZ OEFEESX UDBMETSX AIZ VUEEHZ HMDZSHZO WSFEF AEED TSVE QUZW YMXUZUHSX

EEYEEOPDZSZUFEO MB ZWE FUEZ HMDJ UD TMOHMQ；

（3）次高频 Z 假设由 T 替换得，还原得到：UT QSO VUOHXMOEV GPOTEEVSG TWST OEFEESX UDBMETSX AIT VUEEHT HMDTSHTO WSFEF AEED TSVE QUTW YMXUTUHSX EEYEEOPDTSTUFEO MB TWE FUET HMDJ UD TMOHMQ；

（4）接下来可以先暂停枚举，因为假设不一定正确，如果一味继续向下替换如果构成词句不符合自然语言需要推翻重来。且分析第一个单词 UZ，在此前的假设下 Z 可以替换为 T，那么只需要寻找次高频中哪个字母可以与 T 构成单词且能够放在句首。在剩余次高频字母 A、O、I、N、S、H、R 中，只有 AT 和 IT 能构成词。

（4.1）如果 U 由 A 替换得来，那么密文进一步替换为：AT QSO VAOHXMOEV GPOTEEVSG TWST OEFEESX ADBMETSX AIT VAEEHT HMDTSHTO WSFEF AEED TSVE QATW YMXATAHSX EEYEEOPDTSTAFEO MB TWE FAET HMDJ AD TMOHMQ。但由于英文中很少见到 AET 结尾的单词，因此该替换存疑；

（4.2）如果 U 由 I 替换得来，那么密文进一步替换为：IT QSO VIOHXMOEV GPOTEEVSG TWST OEFEESX IDBMETSX AIT VIEEHT HMDTSHTO WSFEF AEED TSVE QITW YMXITIHSX EEYEEOPDTSTIFEO MB TWE FIET HMDJ ID TMOHMQ；

（5）不断重复上述操作，以频率分析为主，人为剪枝为辅，得到解密后的明文如下：

IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACTS HAVE BEEN MADE WITH POLITICAL REPRESENTATIVES OF THE VIET CONG IN MOSCOW

```
IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACTS HAVE BEEN MADE WITH POLITICAL
REPRESENTATIVES OF THE VIET CONG IN MOSCOW
```

转换表如下：

I : U	Y : G
T : Z	R : E
W : Q	H : W
A : S	V : F
S : O	N : D
D : V	F : B
C : H	M : T
L : X	B : A
O : M	U : I
E : P	P : Y
Y : G	G : J

2. Feistel 加密/解密

Feistel 加密时先将输入明文字符串 M 编码为二进制形式(str 类型)，然后循环加密，最后输出密文 C。Feistel 解密时，先将密文 C 分解为两段，解密后将得到的二进制串还原为字符串。由于 python 中没有二进制数据类型，因此实验过程中必须用到的二进制只能通过字符串形式存储。为了轮中运算方便，在运算中统一使用 int(bin_s,2)函数将二进制串转换成十进制整数进行存储与运算。下面分别介绍主要函数：

（1）密钥生成函数：用于生成每轮用到的子密钥；

```
def Get_Key(K,n):
    '''密钥生成函数
        args:
            K: 密钥(int)
            n: 生成密钥数，即加密轮数(int)
        return:
            Kis: 每轮的密钥(list[str])
    '''
    # 将 int 转换为二进制编码，去掉开头 0b
    K=bin(K)[2:]
    # 计算 Ki 大小
    K_size = len(K)//n
    # 将原始密钥分成多个块
    blocks = [K[K_size*j : K_size*(j+1)] for j in range(n)]
    Kis = []
    for block in blocks:
        Kis.append(block)
    return Kis
```

（2）轮函数：用于加密 R，使用密钥生成函数生成的子密钥；

```
def F(R,K):
    '''对 R 加密的 F 函数，这里使用 SHA256，得到 256 位的二进制数
        args:
            R: 密文(int)
            K: 密钥(int)
        return:
```

```
            bin_R_hashed: 加密后的二进制形式 R 值(str)

    '''

    # 调用 hashlib 模块中的 SHA256 算法，输入需要转换为字节串，得到一个 sha256 哈希对象

    hash_object = hashlib.sha256(str(K).encode('utf-8')+str(R).encode('utf-8'))

    # 哈希摘要将哈希对象压缩为固定长度的字节串

    R_hashed = hash_object.digest()

    # 将字节串转换为二进制串输出

    bin_R_hashed = bin(int.from_bytes(R_hashed, byteorder='big'))

    return bin_R_hashed
```

（3）字符串转二进制串：为了 L 和 R 划分出来的长度相等，将字符串的编码高位用 0 补齐，长度为 8 的整数倍；

```
def str2bin(message):

    '''将字符串转换为对应的二进制编码(str 类型)

        args:

            message: 明文字符串(str)

        return:

            bin_M: 明文字符串的二进制编码(str)

    '''

    # 将输入消息转换为 16 位二进制格式(1byte=8bit，为了 L0 和 R0 能均分，故编码时保留 16 位整数倍)，不足 16 位的在前面补 0

    bin_M = ''.join(format(ord(c), '016b') for c in message)

    # 此处返回的还是 str 类型

    return bin_M
```

（4）二进制串转字符串；

```
def bin2str(bin_M):

    '''将二进制编码(str 类型)转换为字符串

        args:

            bin_M: 二进制编码(str)

        return:

            message: 明文(str)

    '''

    bin_M=bin_M[2:]                          # 去掉二进制串开头的 0b
```

```
    bin_M=bin_M.zfill((len(bin_M)+7)//8*8) # 按 8 位补齐
    message=''
    for i in range(0,len(bin_M),8):
        bin_substr = bin_M[i:i+8]
        n = int(bin_substr,2)
        char = chr(n)
        message += char
    return message
```

（5）Feistel 网络加密函数：先将输入明文字符串 M 编码为二进制形式(str 类型)，然后循环加密，最后输出密文 C；

```
def Feistel_encode(M,K,round=16):
    '''Feistel 网络加密网络
        args:
            M: 明文字符串(str)
            K: 密钥(str 型)
            round: 加密轮数(int)
        return:
            C: 十进制数表示的密文(str)
    '''
    # 将输入消息字符串转换为二进制串
    bin_M = str2bin(M)
    # 分割出 L0 和 R0
    L0,R0 = bin_M[:len(bin_M)//2], bin_M[len(bin_M)//2:]
    # print(len(L0),L0)
    # print(R0)
    # 加密
    L=[int(L0,2)]       # 二进制串转换成数字类型会以十进制 int 形式存储
    R=[int(R0,2)]
    Kis=Get_Key(K,round)
    for i in range(round):
        preR=R[-1]
        preL=L[-1]
        curL=preR
```

```
        curR=preL^int(F(preR,Kis[i]),2)
        L.append(curL)
        R.append(curR)
    # print(L,'\n',R)
    C=str(L[-1])+str(R[-1])
    return C
```

（6）Feistel 网络解密函数：先将密文 C 分解为两段，解密后将得到的二进制串还原为字符串；

```
def Feistel_decode(C,K,round=16):
    '''Feistel 网络解密网络
        args:
            C: 十进制数表示的密文(str)
            K: 密钥(int)
            round: 加密轮数(int)
        return:
            result: 明文字符串(str)
    '''
    # 将密文十进制字符串转换为二进制串
    Ln,Rn = C[:len(C)//2], C[len(C)//2:]
    # 解密
    L=[int(Ln,10)]
    R=[int(Rn,10)]
    Kis=Get_Key(K,round)
    for i in range(round):
        preR=R[-1]
        preL=L[-1]
        curR=preL
        curL=preR^int(F(preL,Kis[round-i-1]),2)
        L.append(curL)
        R.append(curR)
    # 将二进制明文还原为明文字符串
    M=bin2str(bin(int(L[-1])))+bin2str(bin(int(R[-1])))
    return M
```

根据以上函数构建，实现了 Feistel 加密解密文本的功能：

```
if __name__ == '__main__':
    print('M:',end=' ')
    M=input()
    round=16
    K=random.randint(10000000,99999999)
    C_encode=Feistel_encode(M,K,round)
    print('Encoded M:',C_encode)
    M_decode=Feistel_decode(str(C_encode),K,round)
    print('Decoded M:',M_decode)
```

以 CQUINFORMATIONSECURITYEXP 为例：

```
M: CQUINFORMATIONSECURITYEXP
Encoded M: 629323119073918962206256575181230758305464786852700097025604998663977873228979558770401895654830711782694852071269076394053577116613972197651430045418975
6
Decoded M: CQUINFORMATIONSECURITYEXP
```

M 是输入的明文，Encoded M 是加密后的密文，Decoded M 是解密后的明文，可以看到，加密解密后消息信息不变。

五、实验过程中遇到的问题及解决情况

1. 问题：频率分析时一开始单纯比对频率的顺序，但得到的结果甚至无法构成单词；

解决办法：频率分析的字母出现频率顺序只是统计规律，相邻字母可能互换顺序，甚至不符合规律，因此还需结合词义和语义分析，才能够得到正确结果。

2. 问题：频率分析过程中本想通过暴力遍历的方式求解，但复杂度太高，难以在实际应用中得到使用；

解决办法：在频率分析的过程中加上人为认知进行剪枝，可以降低复杂度。

3. 问题：Feistel 网络在生成子密钥时出现非法字符，后来发现是二进制串开头的“0b”；

解决办法：切片只保留数字部分即可：

```
# 将int转换为二进制编码，去掉开头0b
K=bin(K)[2:]
```

4. 问题：调用 hashlib 模块中的 SHA256 函数时出现输入值非法的报错，无法正确计算哈希值；

解决办法：将 int 型变量转换为字节串然后再化为字符串供函数计算：

```
# 调用hashlib模块中的SHA256算法，输入需要转换为字节串，得到一个sha256哈希对象
hash_object = hashlib.sha256(str(K).encode('utf-8')+str(R).encode('utf-8'))
```

5. 问题：在轮函数加密信息时对二进制串 L 和 R 进行异或操作，出现了变量类型错误的问题；

解决办法：python 中没有二进制数据类型，实验呈现结果使用的二进制实际上是只含 0 和 1 的字符串。因此想用其计算得先将 str 化为 int，2 表示按二进制：

```
L=[int(L0,2)]        # 二进制串转换成数字类型会以十进制int形式存储
R=[int(R0,2)]
```

6. 问题：测试字符串长度为奇数时 L 和 R 中切割得到的字符数量不同，在计算过程中出现乱码：

```
M: hello
Encoded M: 939058819736251734316856590664340046730816300749429030346023875734529143141914937531107761790057443199546116308226588871572604384232204369413909588034182 4
Decoded M: V
lo
```

解决办法：将输入消息转换为 16 位二进制格式，不足 16 位的在前面补 0 即可；

六、实验结果及分析和（或）源程序调试过程

使用频度分析法破译密文 UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX AIZ VUEPHZ HMDZSHZO WSFPF APPD TSVP QUZW YMXUZUHSX EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ 得到明文 IT WAS DISCLOSED YESTERDAY THAT SEVERAL INFORMAL BUT DIRECT CONTACTS HAVE BEEN MADE WITH POLITICAL REPRESENTATIVES OF THE VIET CONG IN MOSCOW。实验中并不是强行按照字母出现的统计规律一一对应，而是以频率分析为主，结合人为认知，判断单词的存在性和句子的合法性进行分析。

在实际应用中，为了提高频率分析法的分析效率，通常会先破解出一部分明文，然后根据这部分明文推断出更多的密钥，最终得到完整的明文。频度分析法的破解效果取决于加密算法和密钥长度。对于使用单一替换表的加密算法，密钥长度较短的情况下，频度分析法通常可以有效破解加密文本。但是，对于一些更加复杂的加密算法，如多重替换等，频度分析法的效果会受到影响。

使用 Feistel 加密解密文本 CQUINFORMATIONSECURITYEXP 结果如图：

```
M: CQUINFORMATIONSECURITYEXP
Encoded M: 629323119073918962206256575181230758305464786852700097025604998663977873228979558770401895654830711782694852071269076394053577116613972197651430045418975 6
Decoded M: CQUINFORMATIONSECURITYEXP
```

其中最为重要的就是轮函数 F，需要具备可逆性和混淆性本实验采用的是 SHA256 哈希函数，返回一个 256 位的二进制数。

Feistel 加密时先将输入明文字符串 M 编码为二进制形式(str 类型)，然后循环加密，最后输出密文 C。Feistel 解密时，先将密文 C 分解为两段，解密后将得到的二进制串还原为字符串。Feistel 网络的优点在于它能够使用简单的轮函数来构建高强度的加密算法，而且它的密钥长度也比较短。由于每个部分都使用相同的轮函数，因此它的实现也非常灵活和简单。不过，Feistel 网络也有一些缺点，比如密文的长度与明文长度相同，因此可能会泄露一些信息。