电子科技大学信息与软件工程学院

实 验 报 告

学　　号　2018091618008

姓　　名　袁昊男

（实验）课程名称　现代密码学

理论教师　聂旭云

实验教师　聂旭云

电子科技大学

实 验 报 告

学生姓名：袁昊男　　学号：2018091618008　　指导教师：聂旭云

实验地点：/　　　　实验时间：2020.03.01

一、 实验名称：多表代换密码算法实现

二、 实验学时：/

三、 实验目的：

1、 掌握并实现多表代换密码算法；

2、 掌握密码算法中参数选取、密钥生成、加密和解密的基本流程。

四、 实验原理：

1、 **多表代换密码算法：**首先将明文 M 分为由 n 个字母构成的分组 M_1, M_2, $\cdots$, M_j，对每个分组 M_i 的加密为：

$$C_i \equiv AM_i + B \left(\bmod N \right)，\quad i = 1,2,\cdots,j$$

其中 (A,B) 是密钥，A 是 $n \times n$ 的可逆矩阵，满足 $\gcd\left(\left|A\right|,N\right)=1$（$\left|A\right|$ 是行列式）。$M_i=\left(m_1,m_2,\cdots,m_n\right)^T$，$C=\left(C_1,C_2,\cdots,C_n\right)^T$，$B=\left(B_1,B_2,\cdots,B_n\right)^T$。对密文分组 C_i 的解密为：

$$M_i \equiv A^{-1}\left(C_i - B\right)(\bmod N)，\quad i = 1,2,\cdots,j$$

2、 **逆矩阵：**设 A 是数域上的一个 n 阶矩阵，若在相同数域上存在另一个 n 阶矩阵 B，使得 $AB = BA = I$（注：I 为单位矩阵），则我们称 B 是 A 的逆矩阵（记为 A^{-1}），而 A 则被称为可逆矩阵。

3、 **伴随矩阵：**设矩阵 $A=(a_{ij})_{n\times n}$，将矩阵 A 的元素 a_{ij} 所在的第 i 行第 j 列元素划去后，剩余的各元素按原来的排列顺序组成的 $n-1$ 阶矩阵所确定的行列式称为元素 a_{ij} 的余子式，记为 M_{ij}，称 $A_{ij}=(-1)^{i+j}M_{ij}$ 为元素 a_{ij} 的代数余子式。方阵 $A=(a_{ij})_{n\times n}$ 的各元素的代数余子式 A_{ij} 所构成的如下矩阵

$$A^* = \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{pmatrix}$$

称为 A 的伴随矩阵。

4、 **伴随矩阵法求逆矩阵：**若 n 阶矩阵 A 的行列式不为零，则

$$A^{-1}=\frac{1}{|A|}\cdot A^{*}$$

在模 N 的情况下，上述公式转化为：

$$A^{-1}\equiv|A|^{-1}\cdot A^{*}(\operatorname{mod} N)$$

即：先求出 $|A|$ 在模 N 下的乘法逆元 $|A|^{-1}$（条件：$\gcd(|A|,N)=1$），再将乘法逆元 $|A|^{-1}$ 与伴随矩阵 A^{*} 中的每一个元素进行数乘后模 N。

5、 **例题：**

设 $n=3$，$N=26$，$A=\begin{pmatrix}11&2&19\\5&23&25\\20&7&17\end{pmatrix}$，$B=\begin{pmatrix}0\\0\\0\end{pmatrix}$，明文为“**YOUR PIN NO IS FOUR ONE TWO SIX**”，将明文分成 3 个字母组成的分组“**YOU RPI NNO ISF OUR ONE TWO SIX**”，对应字母表序得：

$$M_1=\begin{pmatrix}24\\14\\20\end{pmatrix},\quad M_2=\begin{pmatrix}17\\15\\8\end{pmatrix},\quad M_3=\begin{pmatrix}13\\13\\14\end{pmatrix},\quad M_4=\begin{pmatrix}8\\18\\5\end{pmatrix},$$

$$M_5=\begin{pmatrix}14\\20\\17\end{pmatrix},\quad M_6=\begin{pmatrix}14\\13\\4\end{pmatrix},\quad M_7=\begin{pmatrix}19\\22\\14\end{pmatrix},\quad M_8=\begin{pmatrix}18\\8\\23\end{pmatrix}。$$

所以：

$$C_1=A\begin{pmatrix}24\\14\\20\end{pmatrix}=\begin{pmatrix}22\\6\\8\end{pmatrix},\quad C_2=A\begin{pmatrix}17\\15\\8\end{pmatrix}=\begin{pmatrix}5\\6\\9\end{pmatrix},\quad C_3=A\begin{pmatrix}13\\13\\14\end{pmatrix}=\begin{pmatrix}19\\12\\17\end{pmatrix},$$

$$C_4=A\begin{pmatrix}8\\18\\5\end{pmatrix}=\begin{pmatrix}11\\7\\7\end{pmatrix},\quad C_5=A\begin{pmatrix}14\\20\\17\end{pmatrix}=\begin{pmatrix}23\\19\\7\end{pmatrix},\quad C_6=A\begin{pmatrix}14\\13\\4\end{pmatrix}=\begin{pmatrix}22\\1\\23\end{pmatrix},$$

$$C_7=A\begin{pmatrix}19\\22\\14\end{pmatrix}=\begin{pmatrix}25\\15\\18\end{pmatrix},\quad C_8=A\begin{pmatrix}18\\8\\23\end{pmatrix}=\begin{pmatrix}1\\17\\1\end{pmatrix}。$$

得密文为“**WGI FGJ TMR LHH XTH WBX ZPS BRB**”。

解密时，先求出矩阵 A 的逆元：

$$A^{-1}=\begin{pmatrix}11&2&19\\5&23&25\\20&7&17\end{pmatrix}^{-1}=\begin{pmatrix}10&23&7\\15&9&22\\5&9&21\end{pmatrix}$$

再求：

$$M_1 = A^{-1}\begin{pmatrix}22\\6\\8\end{pmatrix}=\begin{pmatrix}24\\14\\20\end{pmatrix}，\quad M_2 = A^{-1}\begin{pmatrix}5\\6\\9\end{pmatrix}=\begin{pmatrix}17\\15\\8\end{pmatrix}，\quad M_3 = A^{-1}\begin{pmatrix}19\\12\\17\end{pmatrix}=\begin{pmatrix}13\\13\\14\end{pmatrix}，$$

$$M_4 = A^{-1}\begin{pmatrix}11\\7\\7\end{pmatrix}=\begin{pmatrix}8\\18\\5\end{pmatrix}，\quad M_5 = A^{-1}\begin{pmatrix}23\\19\\7\end{pmatrix}=\begin{pmatrix}14\\20\\17\end{pmatrix}，\quad M_6 = A^{-1}\begin{pmatrix}22\\1\\23\end{pmatrix}=\begin{pmatrix}14\\13\\4\end{pmatrix}，$$

$$M_7 = A^{-1}\begin{pmatrix}25\\15\\18\end{pmatrix}=\begin{pmatrix}19\\22\\14\end{pmatrix}，\quad M_8 = A^{-1}\begin{pmatrix}1\\17\\1\end{pmatrix}=\begin{pmatrix}18\\8\\23\end{pmatrix}。$$

得明文为“**YOU RPI NNO ISF OUR ONE TWO SIX**”。

五、 实验内容：

编程实现多表代换密码算法，要求明文分组或矩阵大小可输入，能够随机生成密钥对输入的英文字母信息进行加密和正确解密。

六、 实验器材（设备、元器件）：

个人 PC 一台。

七、 实验步骤：

1、 学习实验原理，掌握多表代换密码算法。
2、 选择编程语言，设计数据结构，实现多表代换密码算法。
3、 输入不同的明文分组或矩阵大小，对加解密程序进行测试与改进。
4、 总结实验，撰写实验报告。

八、 实验结果与分析（含重要数据结果分析或核心代码流程分析）

1、 密钥生成

(1) 生成随机方阵 $A_{n\times n}$（n 为明文分组或矩阵大小且 $\gcd(|A|,26)=1$）

```python
# 生成随机矩阵
def random(m, n):
    x = np.random.randint(0, 100, (m, n))
    return x


# 扩展欧几里得
def ex_gcd(a, b, arr):
    if b == 0:
        arr[0] = 1
        arr[1] = 0
        return a
    g = ex_gcd(b, a % b, arr)
    t = arr[0]
```

```
    arr[0] = arr[1]
    arr[1] = t - int(a / b) * arr[1]
    return g


# 求矩阵行列式
def mat_det(mat):
    x = np.linalg.det(mat)
    return round(x)


# 生成随即方阵 A 且 det(A)与 26 互素
A = random(n, n)
arr = [0, 1, ]
while ex_gcd(mat_det(A), 26, arr) != 1:
    A = random(n, n)
```

代码说明：使用 Python 作为编程语言，并引入了支持矩阵运算的 NumPy 库。矩阵的生成调用了 NumPy 库的 random()函数，改函数的参数可定义生成矩阵的行数、列数，以及随机生成的矩阵元素大小范围。由于在后续解密步骤中，需要求出$|A|$在模 26 下的乘法逆元$|A|^{-1}$，因此 A 矩阵的行列式与 26 必须互素。这里采用的策略是先随机生成矩阵 A，再判断其行列式是否满足条件：满足则继续；不满足则重新生成矩阵，直至满足为止。矩阵的行列式计算调用了 NumPy 库 linalg 模块的 det()函数，注意返回值时加上 round()四舍五入函数（矩阵中的元素在计算及内部以双精度保存，浮点数计算时可能会出现问题）。采用扩展的欧几里得算法求最大公因数，函数为 ex_gcd()。

(2) 生成随机 n 维向量 B（n 为明文分组或矩阵大小）

```
# 生成随机 B 矩阵
B = random(n,1)
```

2、 加密

```
# 将数字转化为字母
def num_letter(num):
    return chr(num + 64)


# 将字母转化为数字
def letter_num(letter):
    return ord(letter) - 64


# 将矩阵 mod n
def mod_mat(mat, n=26):
    for i in range(mat.shape[0]):
        for j in range(mat.shape[1]):
            mat[i, j] = mat[i, j] % n
    return mat

```

```

# 加密
message = input("请输入明文: ")
new_message = []
for i in message:
    if i == ' ':
        continue
    new_message.append(letter_num(i));
message = new_message
count = len(message)
split = math.ceil(len(message) / n)
C = np.zeros((n, split), dtype = np.int16)
k = 0

for i in range(split):
    for j in range(n):
        if k < count:
            C[j, i] = message[k]
            k = k + 1
        else:
            C[j, i] = 0

for i in range(split):
    C[:, i] = np.matmul(A, C[:, i]) + B.T

C = mod_mat(C, 26)

Enc = []

for i in range(split):
    for j in range(n):
        Enc.append(num_letter(C[j, i]))

Enc_message = "".join(Enc)
print("密文: " + Enc_message)
```

代码说明：此部分需要调用两个函数：num_letter()函数利用 ASCII 码值与字母的映射关系将输入的明文字母转换为字母序（遇到空格时跳过，也可以不跳过，那么在字母与数字的相互转换函数中要对空格字符做特殊映射，使得在解密时能正确输出空格字符）；letter_num()函数是 num_letter()的逆过程；mod_mat()函数将矩阵中每一个元素模 26。加密部分首先遍历明文 list，将字母转换为数字。计算明文的分组数量，调用 math 模块的 ceil()函数对结果向上取整，防止不能恰好分组时造成信息丢失。用两个嵌套的 for 循环将转换后的数字从左至右、从上至下地放入 C 矩阵中，后对 C 矩阵中的每一列进行如下运算：$C_i \equiv AM_i + B \pmod N$。最后将加密后的矩阵元素转换为字母输出，即为密文。

3、 解密

(1) 用伴随矩阵法求 A 的逆矩阵

```
# 求对模 n 的乘法逆元
```

```
def mod_reverse(a, n):
    arr = [0, 1, ]
    gcd = ex_gcd(a, n, arr)
    if gcd == 1:
        return (arr[0] % n + n) % n
    else:
        return -1


# 求伴随矩阵
def adjoint_mat(mat):
    new = np.zeros(mat.shape, dtype=np.int16)
    for i in range(mat.shape[0]):
        for j in range(mat.shape[1]):
            new[j, i] = pow(-1, i + j) * mat_det(cofactor(mat, i, j))

    return mod_mat(new, 26)


# 求代数余子式
def cofactor(self, i, j):
    d = self.shape[0] - 1
    M = np.zeros((d, d), dtype=np.int16)
    for r in range(self.shape[0]):
        if r == i:
            continue
        for c in range(self.shape[1]):
            if c == j:
                continue
            rr = r - 1 if r > i else r
            cc = c - 1 if c > j else c
            M[rr, cc] = self[r, c]
    return M


# 求逆矩阵
def mat_inv(mat):
    arr = [0, 1, ]
    new = np.zeros(mat.shape, dtype=np.int16)
    temp = adjoint_mat(mat)
    det = mat_det(mat)
    if ex_gcd(det, 26, arr) == 1:
        rev = mod_reverse(det, 26)
    new = mod_mat(temp * rev, 26)
    return new
```

代码说明：采用伴随矩阵法求逆矩阵，公式中涉及到了 A 的行列式、A 的伴随矩阵。求行列式函数在前文已经列出，求伴随矩阵则涉及到求矩阵元素的代数余子式：定义函数 cofactor(self, i, j)，参数 self 表示元素所在的原矩阵，i、j 表示元素所在的行与列（从 0 开始），用两个嵌套的 for 循环将除开元素 a_{ij} 所在行、列的剩余元素放入到暂存矩阵 M 中，对 M 求行列式，数值前加上位置符号后放入 new 矩阵的 j 行 i 列位置中（a_{ij} 关于主对角线的对称位置），如此对 A 矩阵的

所有元素进行计算后得到的 new 矩阵即 A 的逆矩阵（返回前对其模 26）。在模 26 下，$1/|A|$转化为$|A|$在模 N 下的乘法逆元$|A|^{-1}$，函数 mod_reverse()采用扩展的欧几里得算法求出乘法逆元。综上，A 的逆矩阵即将 A 的乘法逆元对 A 的伴随矩阵的每一个元素进行数乘后模 26 的结果。

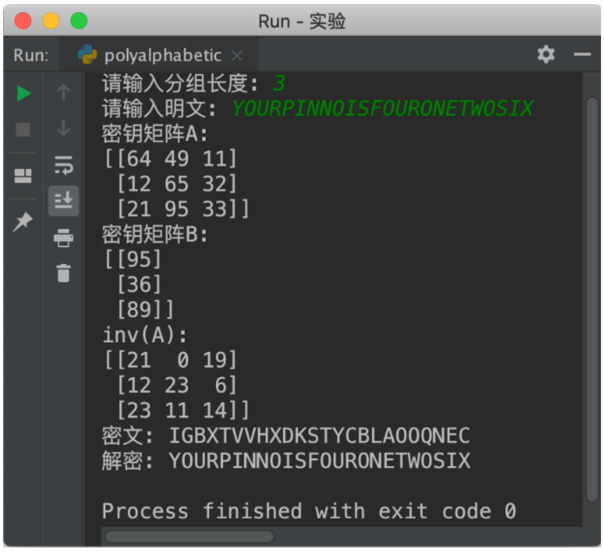
(2) 运算

```python
# 解密
B_ex = np.tile(B, (1, n))
C = C - B
for i in range(split):
    C[:, i] = np.matmul(A_inv, C[:, i])
C = mod_mat(C, 26)

Dec = []
for i in range(split):
    for j in range(n):
        while C[j][i] != 0:
            Dec.append(num_letter(C[j, i]))
            break

Dec_message = "".join(Dec)
print("解密: " + Dec_message)
```

代码说明：调用 NumPy 库中的矩阵乘法函数 matmul()对 C 矩阵进行 $M_i \equiv A^{-1}\left(C_i - B\right)(\bmod N)$运算，得到解密后的矩阵。调用两个嵌套的 for 循环将矩阵中的数字转换为明文字母，当遇到矩阵中的元素为 0 时，结束循环（出现 0 的原因是对明文字符串不能恰好完全分组时，多出来的空余位置以 0 补足）。调用 join()函数将序列中的字符连接生成一个新的字符串输出，即恢复明文。

4、 实验结果

(1) 以 $n=3$ 分组



(2) 以 n =16 分组

```
Run - 多表代替加密
Run:  polyalphabetic  polyalphabetic  polyalphabetic
请输入分组长度: 16
请输入明文: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF
密钥矩阵A:
[[4 4 1 3 6 9 7 8 7 4 9 5 3 9 7 7]
 [2 4 3 9 8 1 8 9 3 4 5 5 9 2 7 7]
 [9 4 2 8 3 4 1 4 7 6 2 4 8 5 8 5]
 [4 7 2 9 6 3 9 7 4 3 2 7 6 2 2 5]
 [1 6 2 8 2 8 7 2 2 4 8 2 6 7 5 9]
 [1 1 9 4 8 9 3 5 1 4 9 4 9 6 5 1]
 [2 8 1 4 9 2 7 8 1 4 1 7 3 1 5 9]
 [4 7 6 2 8 8 6 8 8 3 5 1 5 1 9 6]
 [4 9 2 1 2 5 6 5 4 9 9 3 5 2 6 6]
 [7 9 3 4 1 5 8 6 6 3 8 2 6 7 1 2]
 [5 9 9 7 5 1 4 2 5 3 3 8 6 4 4 1]
 [2 3 1 2 8 6 6 2 2 8 2 2 8 3 9 8]
 [6 3 9 6 1 7 5 2 5 1 6 6 4 9 4 2]
 [3 8 2 5 9 6 3 9 7 4 3 1 5 3 2 4]
 [2 3 4 4 8 9 1 6 5 8 9 3 5 3 3 3]
 [1 3 4 3 5 6 3 8 6 7 4 1 6 5 1 8]]
密钥矩阵B:
[[6]
 [5]
 [6]
 [1]
 [3]
 [6]
 [5]
 [1]
 [3]
 [3]
 [5]
 [7]
 [9]
 [5]
 [7]]
inv(A):
[[19 10 11 23 14  0  0  9 25 11  6 21 15  2 15 17]
 [ 6 11 22 12  3  6 13 18 10  2  6  8 14  7 19  8]
 [25 14  3 13 10  2 10  3 16  9 12 13  6 25  5 13]
 [22 13 25 25  1 10 21 15 20  5 18 23 10  0 19  9]
 [ 9 11 15  5  6  1 13 13 11 20 16  9 11 20  8  6]
 [14 20  4  8 25  7  7  8 25  5  4 22  3 12  5 15]
 [ 2  7  1 22  9  5 17  4  4 10  6 22  5 14 13  1]
 [19 24 12  0 21 25 19 11  2  5 10  9 13 16 22  9]
 [ 8  6 20 14 20  3 12  3  9 18 10 20 11 20  9 19]
 [ 4  9  8 14  5 14 20 22 25  1  2 17 15  4 22 18]
 [11 22  5 24 11 22 12 21 20 25 13  8  1  6 24  2]
 [23 22 13 12  3 24  7 12  9 18 22 10  0 19 16  9]
 [ 5 13  3  7  1  4  5  9  4 23  7 19 10  0 11 10]
 [ 0 18  9  2 16  6 16 13 14  1 16  7 10 14 13  4]
 [ 9 13 16 23  5 18 21  4 22 22 16 19  6 14  1 13]
 [20 22 13  8  9 17 14 20 21 24 24 17 20 18 16 21]]
密文: PXDRNSETSMXIGEZEVVJRDIMHSMRQASTW
解密: ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF

Process finished with exit code 0
```

九、 总结及心得体会：

多表代换加密算法需要用到数论以及线性代数方面的知识。求模整数下的逆矩阵也很有技巧，需要注意每次运算都要模去 26。

多表代换加密及解密算法与 Hill2 加密解密算法大体思路是一致的，区别在于，Hill2 算法是将明文或密文串两两分为一组，使用二阶加密矩阵来进行的加密与解密运算，而多表代换算法则使用三阶加密矩阵，将明文与密文每三个划为一组，来进行加密与解密算法。

Python 中提供了很多与矩阵、数组处理的函数，在一些密码算法的实现上非常方便。

十、 对本实验过程及方法、手段的改进建议：

代码的实现可以写得更简洁明朗一些，逻辑上也应该更加全面。

报告评分：

指导教师签字：