密码学综合设计实验

实验 1：多表代换密码

学号：031803108

姓名：苏煜程

2019 年　9 月　29 日

实验要求

- 根据扩展欧几里得算法实现求乘法逆元模块
- 根据分组长度n, 生成密钥(A,B)。即可逆矩阵A和向量B。生成的密钥要有随机性，密钥以文件形式存储
- 生成A的逆矩阵A-1
- 实现多表代换加密、解密模块
- 实现加密解密软件基本的可视化界面，能够对输入文本框的英文字符串进行加密解密，能够对文本文件中的文本进行加密解密
- 附加内容
    - 对长度不是n的倍数的明文的处理，能够对长度不是n的倍数的明文进行加密解密
    - 对空格和标点符号的处理。要求解密时能还原空格和标点符号

实验原理

多表代换密码实现加密过程主要可以分为以下几个步骤：

- 随机生成n*n的可逆矩阵A，且满足A的行列式与所定模N互素，作为密钥A
- 随机生成n*1的矩阵B，作为密钥B
- 对所需明文就行分组
- 通过 $Ci≡AMi+B(mod\ 26)$ 加密变换求出密文

多表代换密码实现解密过程主要可以分为以下几个步骤：

- 对输入的密钥A作为矩阵A，求出矩阵A的逆矩阵
- 对密文进行分组
- 通过 $Mi≡A-1(Ci-B)(mod\ 26)$ 解密变换求出明文

软件系统设计

随机生成矩阵

此部分主要介绍如何随机生成可逆矩阵A和矩阵B作为密钥A和密钥B，同时实现将密钥作为文件保存

可逆矩阵A

> 生成可逆矩阵A的主要方法为：用单位矩阵进行初等变换而成

生成单位矩阵

```
def getIdentityMatrix(n):#生成单位矩阵
    A = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            if i == j:
                A[i][j] = 1
            else:
                A[i][j] = 0
    return A
```

这没有什么可以讲的很简单

生成随机可逆矩阵A

```
def getInvertibleMatrix(n,m):#生成可逆矩阵
    while 1:
        A = getIdentityMatrix(n)
        tempArray = np.zeros(n)
        B = np.zeros((n,n))
        transformTime = int(random.randint(0,1000))
        maxint = sys.maxsize
        for i in range(transformTime):
            mainRowNum = int((random.randint(0,n) % (n - 1)))#选择一个主行作初等行变
换
            for k in range(n):
                #元素数值是否会溢出
                if maxint - (A[mainRowNum][k])*(int(random.randint(0,10))) < 0
and maxint*(-1) - (A[mainRowNum][k])*(int(random.randint(0,10))) > tempArray[k]:
                    tempArray[k] = A[mainRowNum][k]
                else:
                    tempArray[k] = (A[mainRowNum][k]*
(int(random.randint(0,10))))%m
            for j in range(n):
                if mainRowNum != j:
                    for k in range(n):
                        if maxint - A[j][k]< tempArray[k] and maxint * (-1) -
(A[j][k]) > tempArray[k]:
                            A[j][k] = (A[j][k]/4)%m
                        else:
                            A[j][k] = (A[j][k] + tempArray[k])%m
        if gcd(np.linalg.det(A),m) == 1 and np.linalg.det(A) > 0:
            break
    return A
```

其中`transformTime`为随机决定初等行变换的次数，关键是要确保**矩阵A的行列式与所定模N互素**

故需要此限定的条件：`gcd(np.linalg.det(A),m) == 1 and np.linalg.det(A)`

矩阵B

```
B = np.zeros((n, 1))
for row in range(n):
    B[row] = int(random.randint(0, 26))
```

这没有什么可以讲的很简单

明文分组加密

预处理

记录空格和符号

```
klist = np.zeros(len(clear))
        index = 0
        for i in range(len(clear)):
            if clear[i] == ' ':
                klist[index] = int(i)
                index = index + 1
dlist = np.zeros(len(clear))
index = 0
for i in range(len(clear)):
                if clear[i] == ',':
                dlist[index] = int(i)
            index = index + 1
```

字符串初始化转换为纯字母序列

```
for i in range(len(clear)):
        if clear[i] == ',':
            clear[i] = ' '
    while ' ' in clear:
        clear.remove(' ')
```

补位操作

```
if (lenth % n == 0):
        flag = 0
    else:
        flag = 1
        offset = n - lenth % n
    for i in range(offset):
        CipherText.append('A')
    groups = int(len(CipherText) / n)
```

生成M序列

```
M = np.zeros((groups, n))
    index = 0
    for row in range(groups):
        for column in range(n):
            M[row][column] = CipherText[index]
            index = index + 1
```

分组加密

```
 for i in range(groups):#实现矩阵乘法和矩阵加法，InA*(C-B)
        tmp = np.zeros((n, 1))
        tmpp = np.zeros((n, 1))
        for row in range(n):#初始化Mi
            tmp[row][0] = M[i][row]
        for row in range(n):  # 实现C-B
            tmp[row][0] = tmp[row][0] - B[row][0]
        for row in range(n):#实现InA*(C-B)mod(N)
            for column in range(n):
                tmpp[row][0] = int(tmpp[row][0] + (int(tmp[column][0]) *
int(A[row][column])))
            tmpp[row][0] = int((tmpp[row][0]) % N)
        for row in range(n):
            clearlist.append(NtoA(int(tmpp[row][0])))
```

消除补位影响

```
if flag == 1:
        for i in range(offset):#消除补位影响
            del clearlist[-1]
```

逆矩阵

逆元

```
def InverseElement(b,p):#求逆元
    inv = 1
    while 1:
        if (inv*b)%p == 1:
            break
        inv = inv +1
    return inv
```

行列式

```
def Det(a,n):#求行列式
    b = np.zeros((n, n))#临时数组用于降阶
    sum = int(0)
    sign = int(0)
    p = int(0)
    if n == 1:
        return a[0][0]
    for i in range(n):#此处大循环实现将余子式存入数组b中
        for c in range(n-1):
            for j in range(n-1):
                if c<i:
                    p = 0#当p=0时，行列式只向左移，即消去对应的第一列的数
                else:
                    p = 1#否则行列式左移后再上移
                b[c][j]=a[c+p][j+1]
        if i%2 == 0:
            sign = 1#i为偶数，加法
        else:
            sign = -1#i为奇数，减法
        sum = sum + a[i][0]*Det(b,n-1)*sign#计算行列式的值
```

```
    return  sum
```

伴随矩阵

```
def getAStart(arcs,n,ans):#计算每一行每一列的每个元素所对应的余子式，组成A*
    if n == 1:
        ans[0][0] = 1
    temp = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            for k in range(n-1):
                for t in range(n-1):
                    temp[k][t] = arcs[(k+1 if k>=i else k)][(t+1 if t>=j else
t)]
                ans[j][i] = Det(temp,n-1)
                if (i+j)%2 == 1:
                    ans[j][i] = -ans[j][i]
```

逆矩阵

```
def InverseMat(a,n,p,ans):
    astar = np.zeros((n, n))
    getAStart(a, n, astar)#求A的伴随矩阵
    deta = int(Det(a,n))#求A的行列式
    if deta < 0:
        deta = deta + p
    inv = int(InverseElement(deta,p))
    for i in range(n):
        for j in range(n):
            ans[i][j] = astar[i][j]*inv
            ans[i][j] = ans[i][j]%p
            if ans[i][j] <0:
                ans[i][j] = ans[i][j] + p
```

密文分组解密

补位操作

```
if (lenth % n == 0):
        flag = 0
    else:
        flag = 1
        offset = n - lenth % n
    for i in range(offset):
        CipherText.append('A')
    groups = int(len(CipherText) / n)
```

生成M序列

```
M = np.zeros((groups, n))
    index = 0
    for row in range(groups):
        for column in range(n):
            M[row][column] = CipherText[index]
            index = index + 1
```

分组解密

```
for i in range(groups):#实现矩阵乘法和矩阵加法，InA*(C-B)
        tmp = np.zeros((n, 1))
        tmpp = np.zeros((n, 1))
        for row in range(n):#初始化Mi
            tmp[row][0] = M[i][row]
        for row in range(n):  # 实现C-B
            tmp[row][0] = tmp[row][0] - B[row][0]
        for row in range(n):#实现InA*(C-B)mod(N)
            for column in range(n):
                tmpp[row][0] = int(tmpp[row][0] + (int(tmp[column][0]) *
int(A[row][column])))
            tmpp[row][0] = int((tmpp[row][0]) % N)
        for row in range(n):
            clearlist.append(NtoA(int(tmpp[row][0])))
```

消除补位影响

```
if flag == 1:
        for i in range(offset):#消除补位影响
            del clearlist[-1]
```

GUI图形界面最终实现

GUI的实现使用的**Python**自带的**tkinter**库,实现了错误提示框的弹出和各类操作可视化

```
import random
import tkinter as tk
import numpy as np
from tkinter.simpledialog import *
def gcd(a,b):
    if a%b == 0:
        return b
    else :
        return gcd(b,a%b)
def exgcd(a,b,x,y):
    if b==0:
        x=1
        y=0
    d = exgcd(b,a%b,y,x)
    y = y - (a/b)*x
    return d
def InverseElement(b,p):#求逆元
    inv = 1
    while 1:
        if (inv*b)%p == 1:
            break
        inv = inv +1
    return inv
def Det(a,n):#求行列式
    b = np.zeros((n, n))#临时数组用于降阶
    sum = int(0)
    sign = int(0)
    p = int(0)
    if n == 1:
```

```
        return a[0][0]
    for i in range(n):#此处大循环实现将余子式存入数组b中
        for c in range(n-1):
            for j in range(n-1):
                if c<i:
                    p = 0#当p=0时，行列式只向左移，即消去对应的第一列的数
                else:
                    p = 1#否则行列式左移后再上移
                b[c][j]=a[c+p][j+1]
        if i%2 == 0:
            sign = 1#i为偶数，加法
        else:
            sign = -1#i为奇数，减法
        sum = sum + a[i][0]*Det(b,n-1)*sign#计算行列式的值
    return  sum
def getAStart(arcs,n,ans):#计算每一行每一列的每个元素所对应的余子式，组成A*
    if n == 1:
        ans[0][0] = 1
    temp = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            for k in range(n-1):
                for t in range(n-1):
                    temp[k][t] = arcs[(k+1 if k>=i else k)][(t+1 if t>=j else
t)]
                ans[j][i] = Det(temp,n-1)
                if (i+j)%2 == 1:
                    ans[j][i] = -ans[j][i]
def InverseMat(a,n,p,ans):
    astar = np.zeros((n, n))
    getAStart(a, n, astar)#求A的伴随矩阵
    deta = int(Det(a,n))#求A的行列式
    if deta < 0:
        deta = deta + p
    inv = int(InverseElement(deta,p))
    for i in range(n):
        for j in range(n):
            ans[i][j] = astar[i][j]*inv
            ans[i][j] = ans[i][j]%p
            if ans[i][j] <0:
                ans[i][j] = ans[i][j] + p
    return ans
def AtoN(A):#处理字符和数字的关系
    N = 0
    if(ord(A)>=65 and ord(A)<=90):
        N = ord(A) - ord('A')
    if (ord(A) >= 97 and ord(A) <= 122):
        N = ord(A) - ord('a')
    return N
def NtoA(N):
    A = chr(N + ord('A'))
    return A
def Encrypt(clearlist,A,B,n,N):#加密函数
    CipherText = []
    lenth = len(clearlist)
    flag = 0
    offset = 0
    #补位操作
```

```
    if(lenth%n == 0):
        flag = 0
    else:
        flag = 1
        offset = n-lenth%n
    for i in range(offset):
        clearlist.append('A')
    groups = int(len(clearlist)/n)
    for i in range(len(clearlist)):
        clearlist[i] = AtoN(clearlist[i])
    #生成M序列
    M = np.ones((groups,n))
    index = 0
    for row in range(groups):
        for column in range(n):
            M[row][column] = clearlist[index]
            index = index + 1
    for i in range(groups):#实现矩阵乘法和矩阵加法，A*M+Bmod(N)
        tmp = np.zeros((n, 1))
        tmpp = np.zeros((n, 1))
        for row in range(n):#初始化Mi
            tmp[row][0] = M[i][row]
        for row in range(n):#实现A*M
            for column in range(n):
                tmpp[row][0] = int(tmpp[row][0] + (int(tmp[column][0]) *
int(A[row][column])))
            tmpp[row][0] = int((tmpp[row][0]) % N)
        for row in range(n):#实现+
            tmpp[row][0] = (tmpp[row][0] + B[row][0]) % N
        for row in range(n):
            CipherText.append(NtoA(int(tmpp[row][0])))
    if flag == 1:
        for i in range(offset):#消除补位影响
            del CipherText[-1]
    return CipherText
def Decrypt(CipherText,A,B,n,N):#解密函数
    clearlist = []
    lenth = len(CipherText)
    flag = 0
    offset = 0
    # 补位操作
    if (lenth % n == 0):
        flag = 0
    else:
        flag = 1
        offset = n - lenth % n
    for i in range(offset):
        CipherText.append('A')
    groups = int(len(CipherText) / n)
    for i in range(len(CipherText)):
        CipherText[i] = AtoN(CipherText[i])
    # 生成M序列
    M = np.zeros((groups, n))
    index = 0
    for row in range(groups):
        for column in range(n):
            M[row][column] = CipherText[index]
            index = index + 1
```

```
    #求C-B
    for i in range(groups):#实现矩阵乘法和矩阵加法，InA*(C-B)
        tmp = np.zeros((n, 1))
        tmpp = np.zeros((n, 1))
        for row in range(n):#初始化Mi
            tmp[row][0] = M[i][row]
        for row in range(n):  # 实现C-B
            tmp[row][0] = tmp[row][0] - B[row][0]
        for row in range(n):#实现InA*(C-B)mod(N)
            for column in range(n):
                tmpp[row][0] = int(tmpp[row][0] + (int(tmp[column][0]) *
int(A[row][column])))
            tmpp[row][0] = int((tmpp[row][0]) % N)
        for row in range(n):
            clearlist.append(NtoA(int(tmpp[row][0])))
    if flag == 1:
        for i in range(offset):#消除补位影响
            del clearlist[-1]
    return clearlist
def getIdentityMatrix(n):#生成单位矩阵
    A = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            if i == j:
                A[i][j] = 1
            else:
                A[i][j] = 0
    return A
def getInvertibleMatrix(n,m):#生成可逆矩阵
    while 1:
        A = getIdentityMatrix(n)
        tempArray = np.zeros(n)
        B = np.zeros((n,n))
        transformTime = int(random.randint(0,1000))
        maxint = sys.maxsize
        for i in range(transformTime):
            mainRowNum = int((random.randint(0,n) % (n - 1)))#选择一个主行作初等行变
换
            for k in range(n):
                #元素数值是否会溢出
                if maxint - (A[mainRowNum][k])*(int(random.randint(0,10))) < 0
and maxint*(-1) - (A[mainRowNum][k])*(int(random.randint(0,10))) > tempArray[k]:
                    tempArray[k] = A[mainRowNum][k]
                else:
                    tempArray[k] = (A[mainRowNum][k]*
(int(random.randint(0,10))))%m
            for j in range(n):
                if mainRowNum != j:
                    for k in range(n):
                        if maxint - A[j][k]< tempArray[k] and maxint * (-1) -
(A[j][k]) > tempArray[k]:
                            A[j][k] = (A[j][k]/4)%m
                        else:
                            A[j][k] = (A[j][k] + tempArray[k])%m
        if gcd(np.linalg.det(A),m) == 1 and np.linalg.det(A) > 0:
            break
    return A
def nnb():
```

```
    messagebox.showinfo("Succesfull", "File is in your computer!")
def newwind(n,m):
    winNew = Toplevel(window)
    winNew.geometry('320x400')
    winNew.title('Random Key')
    lb1 = tk.Label(winNew, text='Random A Key', font=('Arial', 16), width=40,
height=2)
    lb1.pack(fill='x')
    nt1 = tk.Text(winNew, height=8, width=40)
    nt1.pack(fill='x')
    lb2 = tk.Label(winNew, text='Random B Key', font=('Arial', 16), width=40,
height=2)
    lb2.pack(fill='x')
    nt2 = tk.Text(winNew, height=8, width=40)
    nt2.pack(fill='x')
    nb = tk.Button(winNew, text='Save', font=('Arial', 12), width=10, height=1,
command=nnb)
    nb.pack()
    lb3 = tk.Label(winNew, text='@FZU-IS-404 ZERO-A-ONE', font=('Arial', 10),
width=40, height=2)
    lb3.pack(fill='x')
    B = np.zeros((n, 1))
    A = getInvertibleMatrix(n,m)
    for row in range(n):
        B[row] = int(random.randint(0, 26))
    At = open('A_key.txt', 'w')
    Bt = open('B_key.txt', 'w')
    for row in range(n):
        str1 = ""
        for line in range(n):
            str1 += str(int(A[row][line]))
            str1 += ' '
        str1 += "\n"
        At.write(str1)
        nt1.insert("%d.%d" % (0, int(row)),str1)
    At.close()
    for row in range(n):
        str2 = ""
        str2 += str(int(A[row][0]))
        str2 += "\n"
        Bt.write(str2)
        nt2.insert("%d.%d" % (0, int(row)), str2)
    Bt.close()
def nb1():
    fo = open("text.txt", "r+")
    str = fo.read()
    tt2.insert(INSERT, str)
    fo.close()
def buttonRK():
    if e1.get() != "" and e2.get() != "":
        n = int(e1.get())
        m = int(e2.get())
        newwind(n,m)
    else:
        messagebox.showinfo("Error", "Please input n")
def n2wind(ans,n):
    n2wind = Toplevel(window)
    n2wind.geometry('300x130')
```

```
    n2wind.title('Inverse Matrix')
    n2nt2 = tk.Text(n2wind, height=8, width=40)
    n2nt2.pack(fill='x')
    for row in range(n):
        str1 = ""
        for line in range(n):
            str1 += str(int(ans[row][line]))
            str1 += ' '
        str1 += "\n"
        n2nt2.insert("%d.%d" % (0, int(row)),str1)
def Dewin():#解密函数GUI
    if t1.get("0.0","end") != "" and t2.get("0.0","end") != "" and e1.get() !=
"" and e2.get() != "":
        n = int(e1.get())
        m = int(e2.get())
        A = np.zeros((n, n))
        B = np.zeros((n, 1))
        InA = np.zeros((n, n))
        Atex = t1.get("1.0", "end")
        stringA = ' '.join(Atex.split())
        listA = list(stringA.split(' '))
        index = 0
        for row in range(n):
            for column in range(n):
                A[row][column] = int(listA[index])
                index = index + 1
        Btex = t2.get("1.0", "end")
        stringB = ' '.join(Btex.split())
        listB = list(stringB.split(' '))
        index = 0
        for row in range(n):
            B[row][0] = int(listB[index])
            index = index + 1
        clear = tt2.get("1.0", "end")
        clear = list(clear)
        InA = InverseMat(A, n, m, InA)
        klist = np.zeros(len(clear))
        index = 0
        for i in range(len(clear)):
            if clear[i] == ' ':
                klist[index] = int(i)
                index = index + 1
        dlist = np.zeros(len(clear))
        index = 0
        for i in range(len(clear)):
            if clear[i] == ',':
                dlist[index] = int(i)
                index = index + 1
        for i in range(len(clear)):
            if clear[i] == ',':
                clear[i] = ' '
        while ' ' in clear:
            clear.remove(' ')
        print(dlist)
        print(klist)
        clearlist = Decrypt(clear, InA, B, n, m)
        print(clearlist)
        clearlist = list(clearlist)
```

```
        for i in range(len(klist)):
            if klist[i] == 0:
                break
            ind = int(klist[i])
            clearlist.insert(ind, " ")
        for i in range(len(dlist)):
            if dlist[i] == 0:
                break
            ind = int(dlist[i])
            clearlist.insert(ind, ",")
        str = ""
        for i in range(len(clearlist)):
            str += clearlist[i]
        t3.insert(INSERT, str)
        n2wind(InA,n)
    else:
        messagebox.showinfo("Error", "Please input n,m,A key,B key")
def Enwin():#加密函数GUI
    if t1.get("0.0","end") != "" and t2.get("0.0","end") != "" and e1.get() !=
"" and e2.get() != "":
        n = int(e1.get())
        m = int(e2.get())
        A = np.zeros((n, n))
        B = np.zeros((n, 1))
        Atex = t1.get("1.0", "end")
        stringA = ' '.join(Atex.split())
        listA = list(stringA.split(' '))
        index = 0
        for row in range(n):
            for column in range(n):
                A[row][column] = int(listA[index])
                index = index + 1
        print(A)
        Btex = t2.get("1.0", "end")
        stringB = ' '.join(Btex.split())
        listB = list(stringB.split(' '))
        index = 0
        for row in range(n):
            B[row][0] = int(listB[index])
            index = index + 1
        print(B)
        clear = tt2.get("1.0", "end")
        clear = list(clear)
        klist = np.zeros(len(clear))
        index = 0
        for i in range(len(clear)):
            if clear[i] == ' ':
                klist[index] = int(i)
                index = index + 1
        dlist = np.zeros(len(clear))
        index = 0
        for i in range(len(clear)):
            if clear[i] == ',':
                dlist[index] = int(i)
                index = index + 1
        print(dlist)
        print(klist)
        for i in range(len(clear)):
```

```
            if clear[i] == ',':
                clear[i] = ' '
        while ' ' in clear:
            clear.remove(' ')
        CipherText = Encrypt(clear,A,B,n,m)
        print(CipherText)
        CipherText = list(CipherText)
        print(CipherText)
        for i in range(len(klist)):
            if klist[i] == 0:
                break
            ind = int(klist[i])
            CipherText.insert(ind," ")
        for i in range(len(dlist)):
            if dlist[i] == 0:
                break
            ind = int(dlist[i])
            CipherText.insert(ind,",")
        str = ""
        for i in range(len(CipherText)):
            str += CipherText[i]
        t3.insert(INSERT,str)
    else:
        messagebox.showinfo("Error", "Please input n,m,A key,B key")
if __name__ == '__main__':
    #主窗口
    window = tk.Tk()
    window.title('Multi-table')
    window.geometry('300x950')
    #随机密钥
    b = tk.Button(window, text='Random Key', font=('Arial', 12), width=10,
height=1,command=buttonRK)
    b.pack()
    # 2标签
    l2 = tk.Label(window, text='Please input n', font=('Arial', 16), width=40,
height=2)
    l2.pack(fill='x')
    # 输入框控件entry
    e1 = tk.Entry(window, show=None, width=8,justify = 'center')  # 显示成明文形式
    e1.pack()
    # 3标签
    l3 = tk.Label(window, text='Please input M', font=('Arial', 16), width=40,
height=2)
    l3.pack(fill='x')
    # 输入框控件entry
    e2 = tk.Entry(window, show=None, width=8, justify='center')  # 显示成明文形式
    e2.pack()
    #4标签
    l4 = tk.Label(window, text='A Key', font=('Arial', 16), width=40, height=2)
    l4.pack(fill='x')
    #Text
    t1 = tk.Text(window, height=8,width = 40)
    t1.pack(fill='x')
    #5标签
    l5 = tk.Label(window, text='B Key', font=('Arial', 16), width=40, height=2)
    l5.pack(fill='x')
    # Text
    t2 = tk.Text(window, height=8, width=40)
```

```
    t2.pack(fill='x')
    # 5标签
    l15 = tk.Label(window, text='Text', font=('Arial', 16), width=20, height=2)
    l15.pack(fill='x')
    # Text
    tt2 = tk.Text(window, height=8, width=20)
    tt2.pack(fill='x')
    b1 = tk.Button(window, text='File', font=('Arial', 12), width=10, height=1,
command=nb1)
    b1.pack()
    # 加密
    De = tk.Button(window, text='Decrypt', font=('Arial', 12), width=10,
height=1,command=Dewin)
    De.pack()
    # 解密
    En = tk.Button(window, text='Encrypt', font=('Arial', 12), width=10,
height=1,command=Enwin)
    En.pack()
    # 6标签
    l6 = tk.Label(window, text='SOLUTION', font=('Arial', 16), width=20,
height=2)
    l6.pack(fill='x')
    # Text
    t3 = tk.Text(window, height=8, width=20)
    t3.pack(fill='x')
    # 8标签
    l8 = tk.Label(window, text='@FZU-IS-404 ZERO-A-ONE', font=('Arial', 10),
width=40, height=2)
    l8.pack(fill='x')
    window.mainloop()
```

重要的实现细节

GUI界面的实现

GUI界面的实现极大的方便了使用者的使用体验，降低了使用门槛，使得本程序的实用化程度大大提升

三目运算符的使用

在求A的伴随矩阵时，使用了三目运算符，简化了程序，使思路更清晰

```
temp[k][t] = arcs[(k+1 if k>=i else k)][(t+1 if t>=j else t)]
```

随机生成在模下可逆矩阵

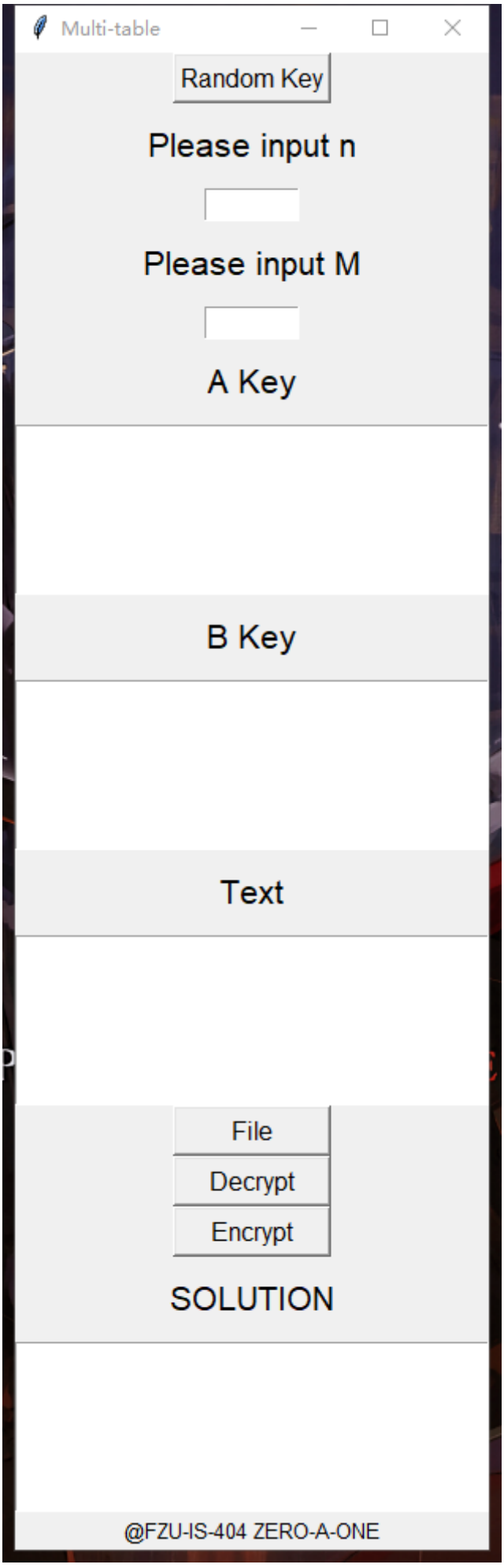
通过模拟初等变换实现的可逆矩阵，更重要的是保证了矩阵A的行列式与所定模N互素。且在随机化的过程中，通过判断保证了数据不会溢出

```
if maxint - (A[mainRowNum][k])*(int(random.randint(0,10))) < 0 and maxint*(-1) -
(A[mainRowNum][k])*(int(random.randint(0,10))) > tempArray[k]:
```

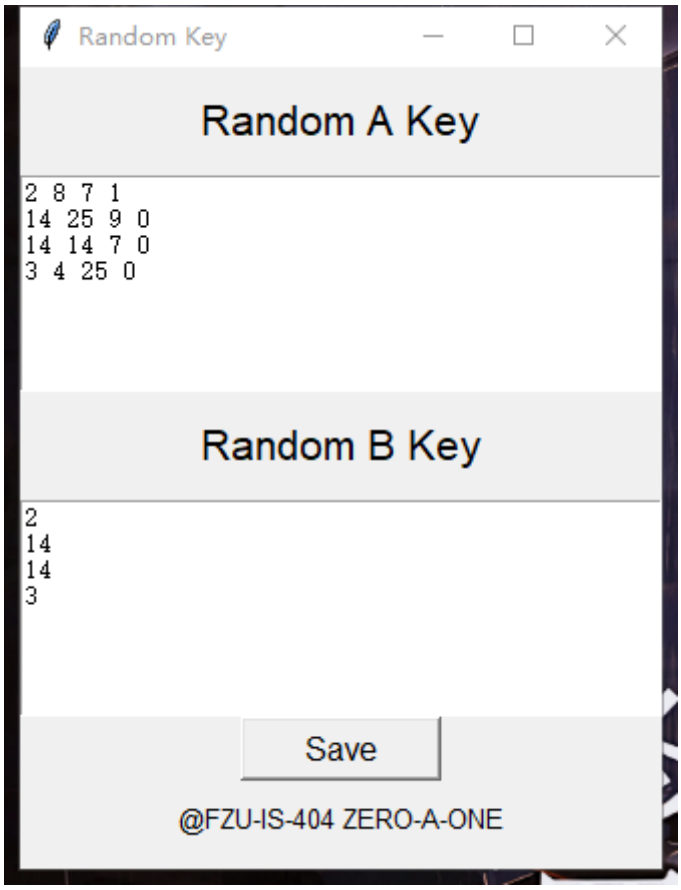
`

实现效果

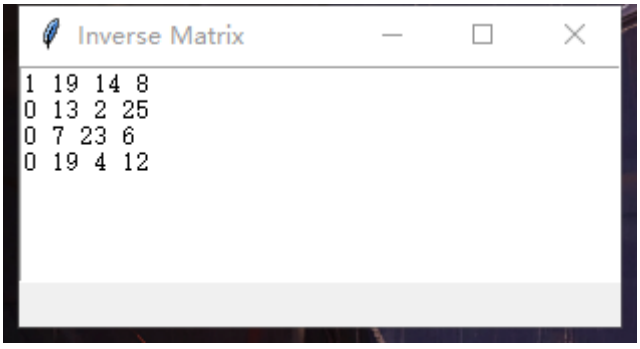
主界面


Multi-table
Random Key
Please input n
Please input M
A Key
B Key
Text
File
Decrypt
Encrypt
SOLUTION
@FZU-IS-404 ZERO-A-ONE

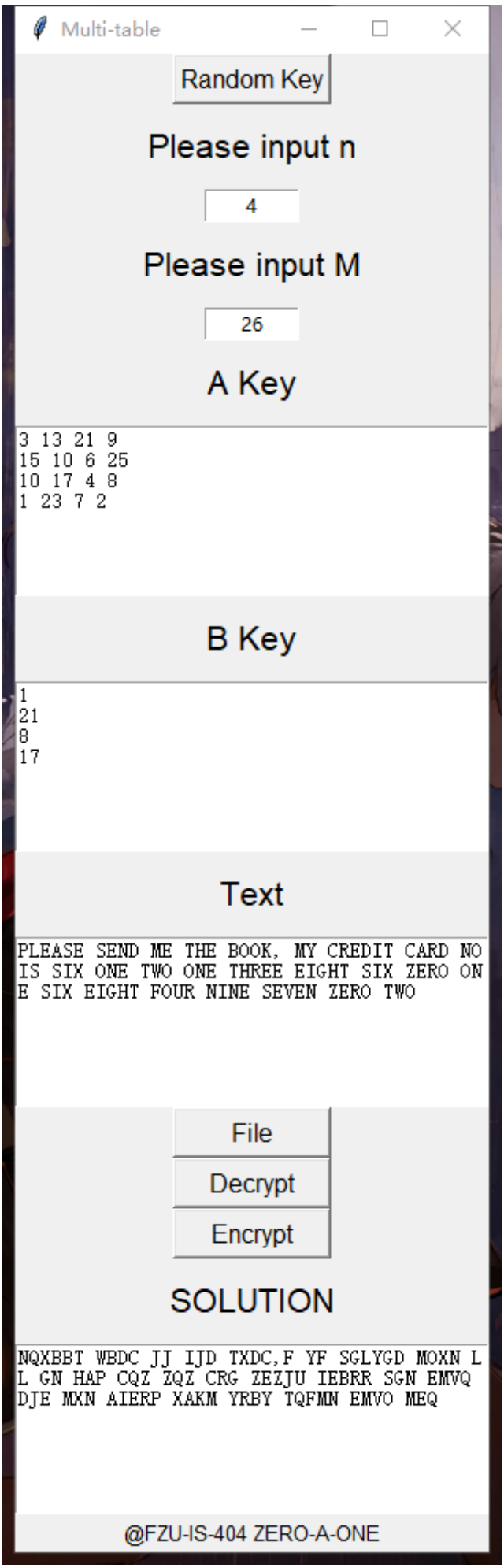
随机矩阵生成



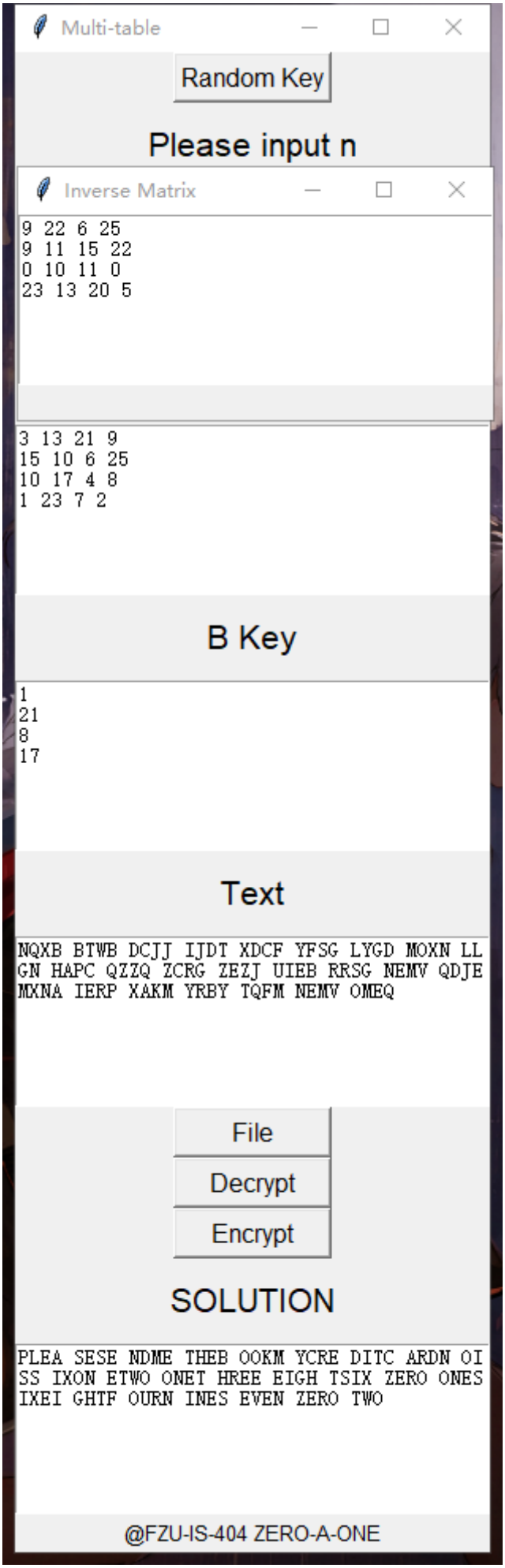
逆矩阵



例题加密


Multi-table
Random Key
Please input n
4
Please input M
26
A Key
3 13 21 9
15 10 6 25
10 17 4 8
1 23 7 2
B Key
1
21
8
17
Text
PLEASE SEND ME THE BOOK, MY CREDIT CARD NO
IS SIX ONE TWO ONE THREE EIGHT SIX ZERO ON
E SIX EIGHT FOUR NINE SEVEN ZERO TWO
File
Decrypt
Encrypt
SOLUTION
NQXBBT WBDC JJ IJD TXDC,F YF SGLYGD MOXN L
L GN HAP CQZ ZQZ CRG ZEZJU IEBRR SGN EMVQ
DJE MXN AIERP XAKM YRBY TQFMN EMVO MEQ
@FZU-IS-404 ZERO-A-ONE

例题解密


Multi-table
Random Key
Please input n
Inverse Matrix
9 22 6 25
9 11 15 22
0 10 11 0
23 13 20 5
3 13 21 9
15 10 6 25
10 17 4 8
1 23 7 2
B Key
1
21
8
17
Text
NQXB BTWB DCJJ IJDT XDCF YFSG LYGD MOXN LL
GN HAPC QZZQ ZCRG ZEZJ UIEB RRSG NEMV QDJE
MXNA IERP XAKM YRBY TQFM NEMV OMEQ
File
Decrypt
Encrypt
SOLUTION
PLEA SESE NDME THEB OOKM YCRE DITC ARDN OI
SS IXON ETWO ONET HREE EIGH TSIX ZERO ONES
IXEI GHTF OURN INES EVEN ZERO TWO
@FZU-IS-404 ZERO-A-ONE

总结

多表代换密码为古典密码学中一种较为经典的加密方式，对于多表替换加密来说，加密后的字母几乎不再保持原来的频率，对于词频和字频分析有了一定的抵抗能力

本人的系统有如下亮点：

- 实现了全部的GUI可视化操作
- 实现了对长度不是n的倍数的明文的处理，能够对长度不是n的倍数的明文进行加密解密
- 实现了对空格和标点符号的处理。要求解密时能还原空格和标点符号
- 实现了对文件的读写操作
- 实现了调用系统API实现错误弹窗功能