**TABLE OF CONTENTS**

[1. INTRODUCTION 1](#_Toc501351469)

[2. LITERATURE REVIEW 2](#_Toc501351470)

[2.1 Preliminary 2](#_Toc501351471)

[2.2 The Construction of Generalized Fibonacci Matrix 2](#_Toc501351472)

[2.4 Construction of pq-Generalized Fibonacci Matrix 6](#_Toc501351473)

[3. TECHNOLOGY AND METHODS 8](#_Toc501351474)

[3.1 System Requirements 8](#_Toc501351475)

[3.1.1 Hardware Requirements 8](#_Toc501351476)

[3.1.2 Software Requirements: 8](#_Toc501351477)

[3.2 Method 8](#_Toc501351478)

[3.2.1 Encryption 8](#_Toc501351479)

[3.2.2 Verification 8](#_Toc501351480)

[3.2.3 Decryption 9](#_Toc501351481)

[3.3 Implementation of Encryption/Decryption in Python 9](#_Toc501351482)

[3.3.1 Encryption 9](#_Toc501351483)

[3.3.2 Verification 9](#_Toc501351484)

[3.3.3 Decryption 10](#_Toc501351485)

[4. CONCLUSION 11](#_Toc501351486)

[5. RESULTS AND DISCUSSION 12](#_Toc501351487)

[5.1 Encryption/Decryption Module 12](#_Toc501351488)

[5.2 GUI 14](#_Toc501351489)

[6. REFERENCES 16](#_Toc501351490)

[APPENDIX 17](#_Toc501351491)

# INTRODUCTION

Fibonacci proposed a sequence {Fn}:

This is the famous Fibonacci sequence, here every term*Fn*is called Fibonacci number. Both Fibonacci numbers andFibonacci sequence have important effects on combinatorialmathematics, probability statistics, graph theory, operationsresearch, etc. In recent years, along with the development ofinformation technology, they have been widely used in the fieldof information security.Many extensive researches have been made on Fibonaccinumber and Fibonacci sequence, and various research methods have been proposed. Hogga proposed Fibonacci *Q*Matrixto offer matrix tools for studying Fibonacci numbers. ErKaraduman expanded matrix representation and proved thatthe sum of generalized Fibonacci numbers can be derived usingthis representation. In this project, firstly, we give *LDU* triangular decompositionform of the generalized Fibonacci matrix. Then by this form wepropose a method to construct the generalized Fibonacci matrixand gain its equivalent classification. Secondly, we extend ouridea to *pq*-generalized Fibonacci matrix and give itsconstruction method. Finally, an encryption/decryptionalgorithm based on *pq*-generalized Fibonacci matrix is proposed for secure text chat which has verification capability at the receiver’s end.

# LITERATURE REVIEW

## 2.1 Preliminary

**2.1.1 Definition:** If **Fn = F n-1 + F n-2**, then sequence Fn is called pq-Fibonacci sequence, also written by .



**2.1.2 Definition:** Thematrix is called Fibonacci Matrix. All the sequential principal minors of an Fibonacci matrix make a Fibonacci sequence.

**2.1.3 Definition:**If the sequential principal minors of an *n* x*n* matrix *A* make a Fibonacci sequence, then we call a generalized Fibonacci matrix.

**2.1.3 Definition:**The matrix whose sequential principal minors making a *pq*-Fibonacci sequence is called *pq*generalized Fibonacci matrix.

**2.1.4 Definition:**If nonsingular matrix can be written as the product of a lower triangular matrix and a upper triangular matrix , namely , then we call *A* having triangular factorization.

## 2.2 The Construction of Generalized Fibonacci Matrix

**2.2.1 Lemma:** Let *A* be a matrix, *A* can be uniquely written as if and only if the sequential

principal minors of *A* satisfy , where *L* is unit lower triangular matrix, *U* is unit upper triangular matrixand is a diagonal matrix. Here

For simplicity, we denote a diagonal matrix by , and an *n* x *n* diagonal matrix by .

**2.2.2 Lemma:** Suppose that is a nonsingularsquare matrix, then

=

**2.3.3 Theorems:**

**Theorem 1:** (i) Any generalized Fibonacci matrix can certainly be uniquely written as , where is unit lower triangular matrix, is unit upper triangular matrix,

and is Fibonacci sequence.

(ii) If , where , and are defined in Theorem 1 (i), thenis generalizedFibonacci matrix.

**Proof:**It can be directly obtained by Lemma 1. We only need to prove that sequential principal minors of *A* satisfy.

Let

By Lemma 2, we have

Here, , and

So, . As a result, *A* is generalizedFibonacci matrix.

**Theorem 2:**(i) The matrix in the form of is generalized Fibonacci matrix, where *L* is unit lower triangularmatrix and

(ii) Any generalized Fibonacci matrix *A* mustbe congruent with generalized Fibonacci matrix in the form of. Specially, there exists an unit lowertriangular matrix such that, and P is unique.

**Proof:** When , let

And

Then

If the above equation can be written in the form of, then it need to satisfy that

That is to say, , namely, .

Let

we have thatand that *P* is unique

For arbitrary positive integer supposing that theproposition holds when , now we consider the case of .

Let

Where and According to the supposition, for generalized Fibonacci matrix there exists unit lower triangular matrix such that satisfies

… ( 1 )

Let

Where then

If the proposition holds, then we have

…(2)

*… ( 3 )*

Equation (2) obviously holds by (1). Next, we only need toconsider (3). Namely,

…( 4 )

From the definitions ofand we have

Namely,

Due to that is invertible matrix, the system of (5) issolvable (i.e., there exists such that (3) holds), and thus *P*exists. So, any n x ngeneralized Fibonacci matrix is bound tobe congruent with generalized Fibonacci matrix in the form of*LD(2I-)* . Furthermore, according to the uniqueness ofsolutions, is uniquely determined. In addition, is unique.Therefore, the unit lower triangular matrix *P* by which ageneralized Fibonacci matrix is congruent with the matrix inform of *LD*(2I-) is solely determined.

From the above discussions, arbitrary generalized Fibonaccimatrix of order *n* is certainly congruent with generalizedFibonacci matrix in the form of*LD*(2I-). Conversely, forany two generalized Fibonacci matrices and, where 1 *L* and 2 *L* are unit lowertriangular matrix of order *n*. Due to the uniqueness offactorization in Lemma 1, there holds *A* ≠*B* if .

**Remark:**In fact, the first part proof of Theorem 2 providesa common method to construct generalized Fibonacci matrix.Next, we consider the equivalence classification problem of

generalized Fibonacci matrix.

Denote

Φ={all generalized Fibonacci matrices of order *n*};

Ψ={ all unit lower triangular matrices of order *n*}.

Let Φ, then Ǝ P,QΨ, such that

.

.

Define that

It is easy to verify that “” forms equivalence relationship,thus one person may obtain equivalence classification of \_ bythe relationship “”. As a result, the equivalence class of *A*may be defined as follows:

## 2.4 Construction of pq-Generalized Fibonacci Matrix

**Theorem 3**:

(i) Given *pq*-Fibonacci sequence pq-{},

=

let

then for arbitrary unit lower triangular matrix *L* of order *n*, wehave that is *pq*-generalized Fibonacci matrix.

(ii) If *pq*-generalized Fibonacci matrix can bewritten as, then matrix equation

F =

has a unique solution nxn. Furthermore, the solution canbe obtained by the iterative formulas below:

**Proof:**

**(i)**It can be obtained similar to the proof of Theorem 1.

**(ii)** Note that





=

By comparison, the above iterative formulas can be directly obtained.

# TECHNOLOGY AND METHODS

## 3.1 System Requirements

### 3.1.1 Hardware Requirements

1. **Processor:** Intel Pentium 4 or better.
2. **RAM:** 1 GB minimum.
3. **Peripherals:** Keyboard, Mouse.

### 3.1.2 Software Requirements:

1. **O/S:** Ubuntu 16.04 or above, Windows 8 or above.
2. **Environment:** Anaconda

## 3.2 Method

### 3.2.1 Encryption

1. Divide the whole plaintext into blocks of
2. Put every plaintext block into unit lower triangular matrix, written by *M*;
3. In encryption, compute the corresponding ciphertext block and send *C* to the receiver;
4. Verify whether *C* is *pq*-generalized Fibonacci matrix or not. If not, the receiver requests for repetition.

**Example:**Taking *n* = 3 as example, the sender and receiverchoose *p* =*q =*1 . Suppose the original information {*a*,*b*,*c*} begiven in the form of

then the ciphertext

### 3.2.2 Verification

After the receiver obtained *C*, he(she) computes threesequential principal minors as follows:

* ,
* ,

From the above calculated principal minors of the matrix, we have and as 1, 1, 2 respectively. Since and are Fibonacci sequence, the matrix received is a pq-generalized Fibonacci matrix.

### 3.2.3 Decryption

In decryption step, we recover *M* from *C* according to theiterative formulas in Theorem 3 as:

* ,
* ,

Where p and q are supplied keys (i.e. 1 and 1 respectively).

Where , and are the three elements on the lower triangle of the matrix *M* containing the plaintext. Hence, we recovered the plaintext *abc.*

## 3.3 Implementation of Encryption/Decryption in Python

Following code snippets are Python3 implementations of the above method using numpy:

### 3.3.1 Encryption

M = np.matrix('1 0 0; '+ str(a) +' 1 0; '+ str(b) +' '+ str(c) +' 1', dtype=object)

I = np.matrix('1 0 0; 0 1 0; 0 0 1', dtype = object)

D = np.matrix(str(p) +' 0 0; 0 ' + str(q) + ' 0; 0 0 ' + str(r), dtype=object)

X = (2\*I - M.transpose())

C = M \* D \* X

### 3.3.2 Verification

mList = M.tolist()

m1List = [mList[0][0]]

m2List = [[mList[0][0], mList[0][1]],[mList[1][0], mList[1][1]]]

m3List = mList

d1 = np.linalg.det(np.matrix(m1List))

d2 = np.linalg.det(np.matrix(m2List))

d3 = np.linalg.det(np.matrix(m3List))

return (d1 + d2) == d3

### 3.3.3 Decryption

pInv = -1/p

a = int(round(pInv\*M[0,1]))

b = int(round(pInv\*M[0,2]))

c = int(round(-1\*p\*a\*b - M[1,2])/q)

return chr(a) + chr(b) + chr(c)

# CONCLUSION

In the field of communications, a cryptographic algorithm with verification capability is desirable. In this project, a list of *pq*-generalized Fibonacci matrices are constructed after dividing the plaintext into groups of three characters and constructing the *pq*-generalized Fibonacci matrices of each group of characters in sequence. In addition, the decryption algorithm receives the list of *pq*-generalized Fibonacci matrices and decrypts each matrix iteratively in the order of their sequence as serialized by the encryption algorithm. Compared with the most existing cryptographic algorithms, the algorithm has the merits of gaining verification capability and large key amount.

In last semester, the encoding/decoding module has been completed along with basic CLI server/client implementation and some GUI dialogs.

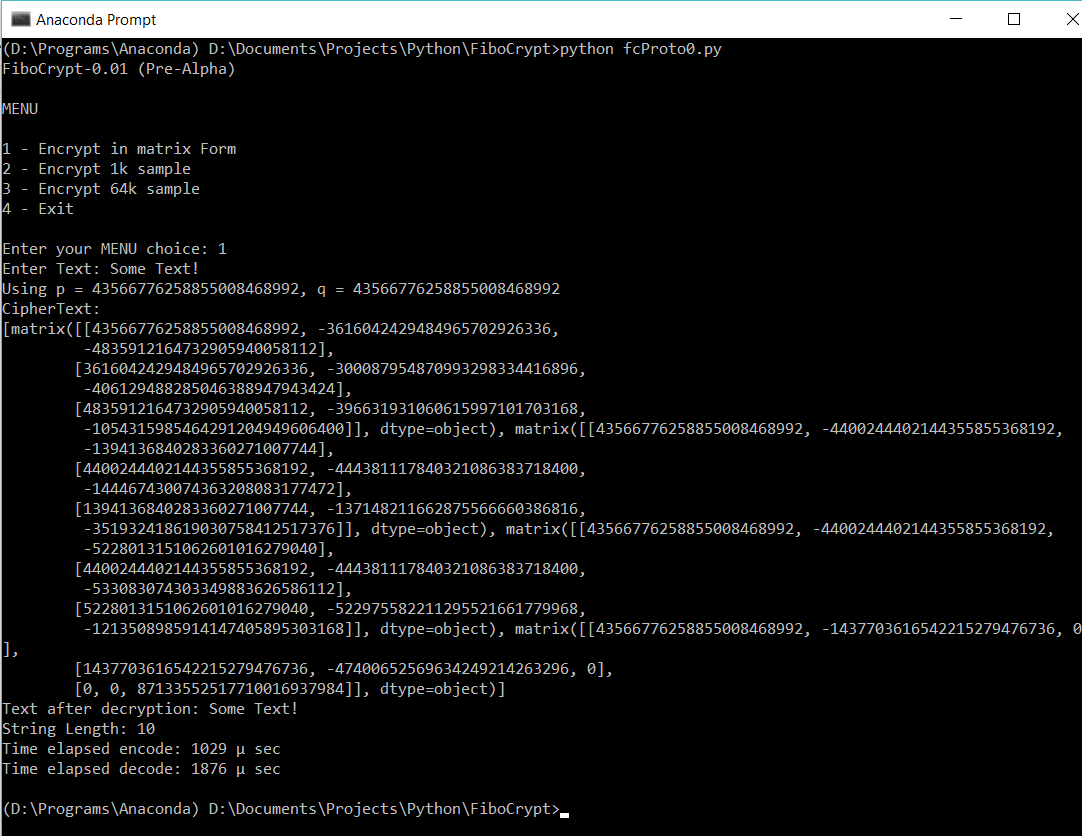
In this semester, user registration on the server has been done along with basic chat functionality. The remaining task is to integrate them in order to enable the registered user to chat with registered user.

# RESULTS AND DISCUSSION

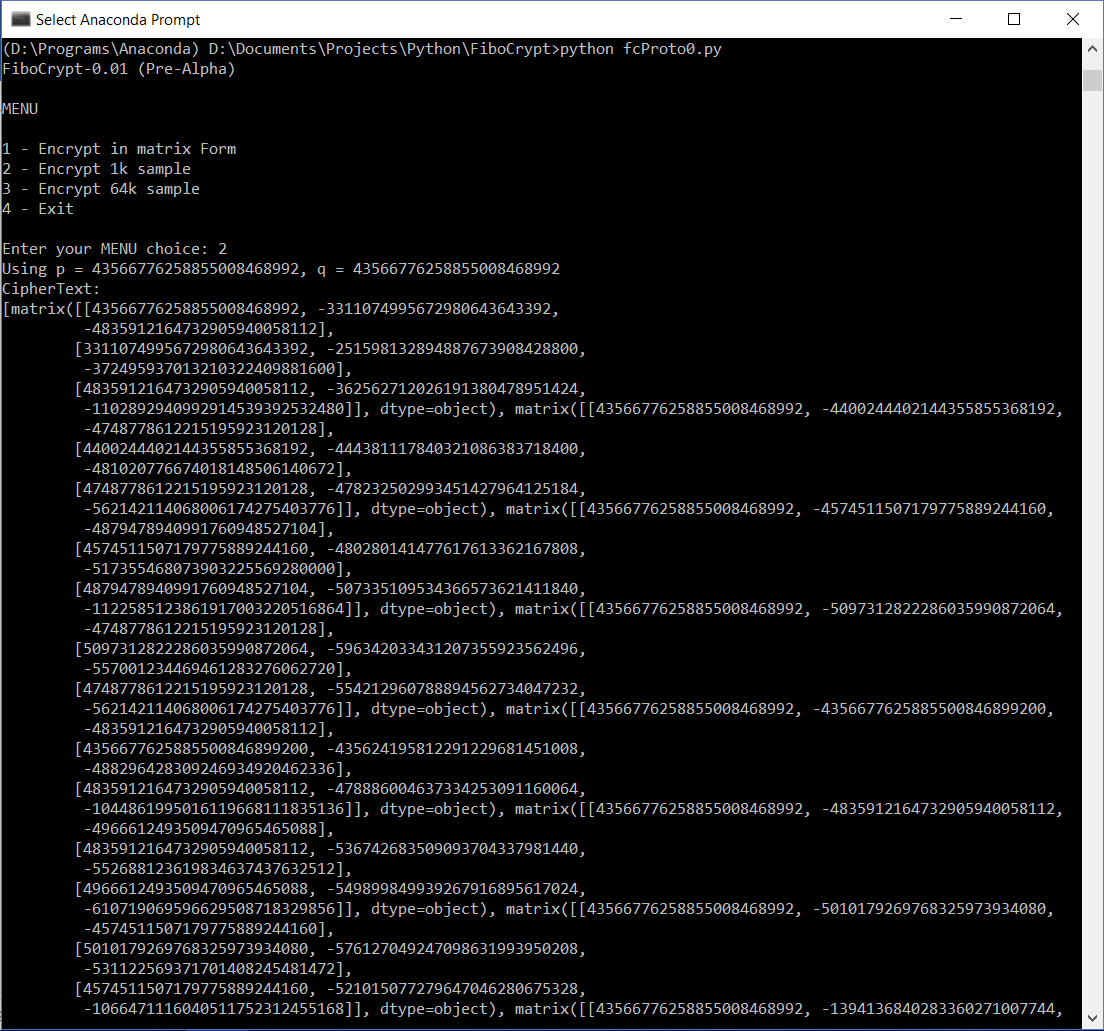
## 5.1 Encryption/Decryption Module

The pq-Generalized matrix method for encryption is secure as well as fast method of encryption and decryption. It also has capability to support infinite amount of keys.

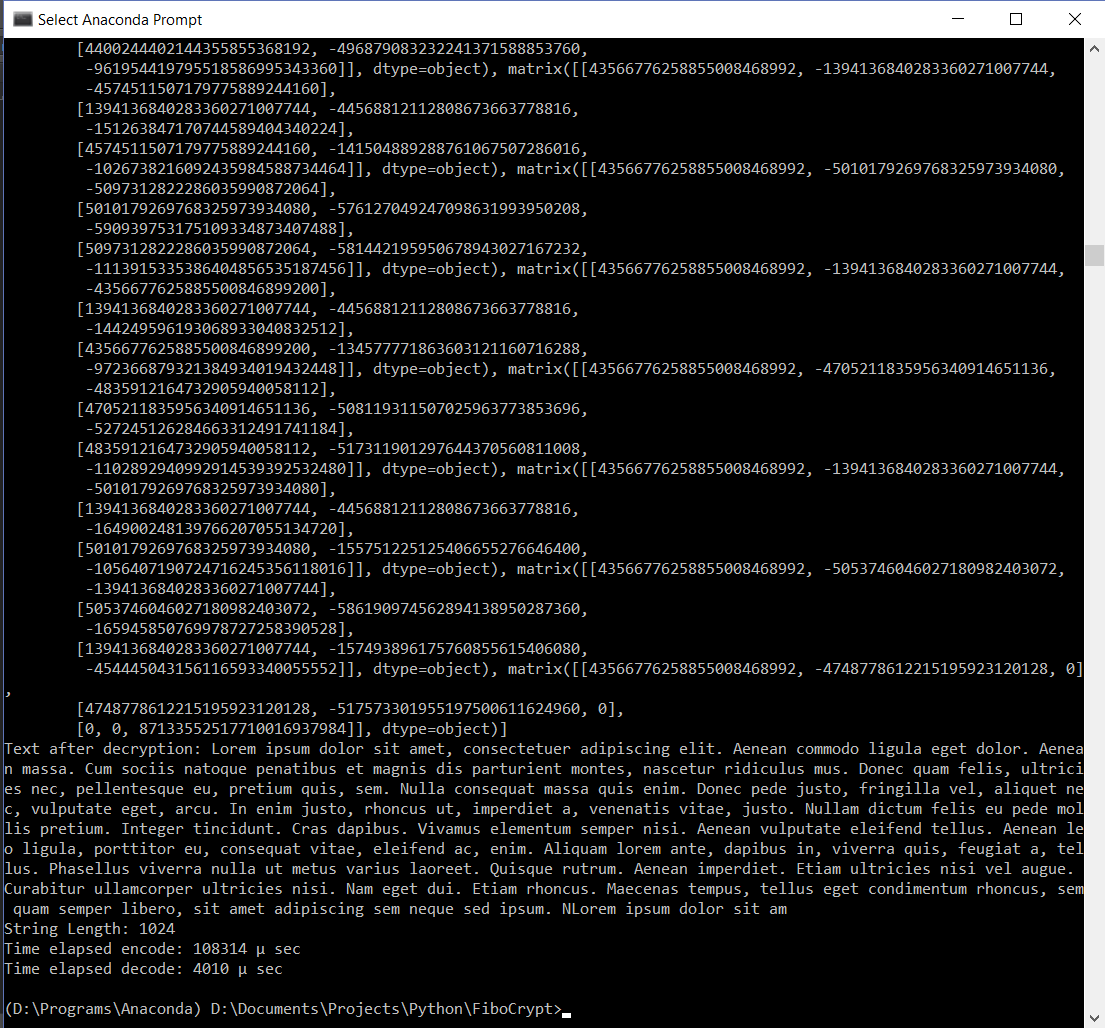
The following screenshots show the performance of the algorithm with keys: p = 43566776258855008468992 and q = 43566776258855008468992 which are big integers and are both part of the Fibonacci series.

****

**Snapshot1:** Execution for a sample text: “Some Text!”

****

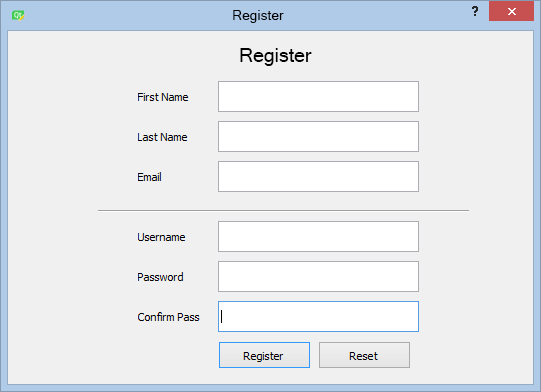
**Snapshot2:** The encryption/decryption of 1k char text (Part 1)



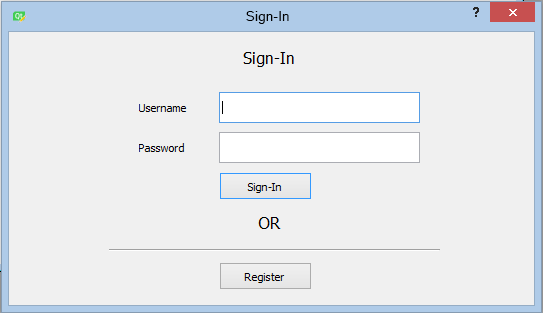
**Snapshot3:** The encryption/decryption of 1k char text (Part 1)

## 5.2 GUI

Following are the snapshots of the GUI



**Snapshot4:** The Register Dialog



**Snapshot5:** The Sign-In Dialog

# REFERENCES

|  |  |
| --- | --- |
| [1] | Y. P. Cheng, Matrix theory, Xi’an: Northwest Industrial University press, 2001. |
| [2] | S. W. Bai, Selected topics in advanced algebra, Harbin: Heilongjiang Education Press, 1996. |
| [3] | Federal information Processing Standrd(FIPS), Data Encryption Standard, Nat. Bur. Stand, 1977. |
| [4] | Advanced Encryption Standard, National Institute of Standards and Technology (FIPS), 2001. |
| [5] | G. J. Simmons, "Authentication theory/coding theory," *Advances in cryptology proceedings of Crypto’84,* no. 196, pp. 411-432, 1985. |
| [6] | X. Z. X. D. Fu, "On matrices related with Fibonacci and Lucas numbers," *Applied mathematics and computation,* no. 200, pp. 96-100, 2008. |
| [7] | N. J. a. S. I. S. Predrag, "A generalization of Fibonacci and Lucas matrices," *Discrete applied mathematics,* no. 156, pp. 2606-2619, 2008. |
| [8] | V. E. Hoggatt, Fibonacci and Lucas numbers, Boston: Houghton-Mifflin, 1969. |
| [9] | E. Karaduman, "Sums of Fibonacci numbers by matrix methods," *The Fibonacci Quarterly,* vol. 3, no. 22, 1984. |
| [10] | X. H. C. J. L. Chen, Special matrices, Beijing: Tsinghua University Press, 2000. |

# APPENDIX

**[A] Code for /fcProto0.py**

import numpy as np

from random import random

from datetime import datetime

fibArr = []

fibArrInit = False

def initFibArr(n = 20):

global fibArr

global fibArrInit

fibArrInit =True

p = 1

q = 1

for i in range(n):

tmp = q

q += p

p = tmp

fibArr.append(p+q)

def fiboCryptUnicode(text, p, q):

mList = fibocrypt(text, p, q)

leftMask = int('11111111111111110000000000000000', 2)

rightMask = int('00000000000000001111111111111111', 2)

uniText = u''

for matrix in mList:

for i in range(matrix.shape[0]):

for j in range(matrix.shape[1]):

num = matrix[i, j]

left = (num&leftMask) >> 16

right = num&rightMask

uniText += chr(left)+chr(right)

print(str(num) + '=' + chr(left)+chr(right))

print(bin(num) + '=' + bin(left)+bin(right)+'||'+bin((left<<16)|right)+'|'+str((left<<16)|right))

uniText += u',' + chr(int(random()\*10000))

return uniText

def fibocrypt(text, p, q):

global fibArrInit

if not fibArrInit:

initFibArr()

r = p+q

txtLen = len(text)

rem = txtLen % 3

divF = txtLen//3

if rem != 0:

divF += 1

for i in range(rem):

text += '\0'

txtLen = len(text)

## The list of matrices containtingCryptText

matList = []

for i in range(0, txtLen - 1, 3):

try:

a = ord(text[i])

except(ValueError, IndexError):

a = 0

try:

b = ord(text[i+1])

except(ValueError, IndexError):

b = 0

try:

c = ord(text[i+2])

except(ValueError, IndexError):

c = 0

M = np.matrix('1 0 0; '+ str(a) +' 1 0; '+ str(b) +' '+ str(c) +' 1', dtype=object)

I = np.matrix('1 0 0; 0 1 0; 0 0 1', dtype = object)

D = np.matrix(str(p) +' 0 0; 0 ' + str(q) + ' 0; 0 0 ' + str(r), dtype=object)

X = (2\*I - M.transpose())

C = M \* D \* X

matList.append(C)

return matList

def decryptList(mList, p, q):

text = ''

for i in mList:

text += deFibo(i, p, q)

return text

def decryptUnicode(text, p, q):

negMask = 1 << 31

textLen = len(text)

mList = []

currList = []

currElemCount = 0

for i in range(0, textLen, 2):

if text[i] == ',':

rank = 3

temp = []

for j in range(rank):

temp\_ = []

for k in range(rank):

temp\_.append(currList[((j\*rank) + k)])

temp.append(temp\_)

matrix = np.matrix(temp, dtype=object)

mList.append(matrix)

currList = []

currElemCount = 0

left = ord(text[i]) << 16

right = ord(text[i+1])

num = left | right

currList.append(num)

currElemCount += 1

return decryptList(mList, p, q)

def verify(M):

mList = M.tolist()

m1List = [mList[0][0]]

m2List = [[mList[0][0], mList[0][1]],[mList[1][0], mList[1][1]]]

m3List = mList

d1 = np.linalg.det(np.matrix(m1List))

d2 = np.linalg.det(np.matrix(m2List))

d3 = np.linalg.det(np.matrix(m3List))

return (d1 + d2) == d3

def deFibo(M, p, q):

dim = M.shape

if dim[0] != dim[1]:

print('error! the matrix is not square!')

return None

pInv = -1/p

a = int(round(pInv\*M[0,1]))

b = int(round(pInv\*M[0,2]))

c = int(round(-1\*p\*a\*b - M[1,2])/q)

return chr(a) + chr(b) + chr(c)

def main():

print('FiboCrypt-0.01 (Pre-Alpha)\n')

print('MENU\n')

print('1 - Encrypt in matrix Form')

print('2 - Encrypt 1k sample')

print('3 - Encrypt 64k sample')

print('4 - Exit\n')

menuResponse = int(input('Enter your MENU choice: '))

initFibArr()

time2 = None

if menuResponse == 1:

text = input('Enter Text: ')

if menuResponse == 2 or menuResponse == 3:

text = ''

if menuResponse == 2:

with open('text1k.txt') as f:

for line in f:

text += line

if menuResponse == 3:

with open('text64k.txt') as f:

for line in f:

text += line

p = 43566776258855008468992

q = 43566776258855008468992

print('Using p = ' + str(p) + ', q = ' + str(q))

time1 = datetime.now()

cryptList = fibocrypt(text, p, q)

time2 = datetime.now()

diff1 = time2 - time1

print('CipherText:\n'+str(cryptList))

time1 = datetime.now()

print('Text after decryption: ' + decryptList(cryptList, p, q))

time2 = datetime.now()

diff2 = time2 - time1

print('String Length: ' + str(len(text)))

print('Time elapsed encode: ' + str(diff1.microseconds) + ' \u03bc sec')

print('Time elapsed decode: ' + str(diff2.microseconds) + ' \u03bc sec')

if \_\_name\_\_ == '\_\_main\_\_':

main()

**[B] Code for /server.py**

import socket

import select

import sys

from thread import \*

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

IP\_address = str(sys.argv[1])

Port = int(sys.argv[2])

server.bind((IP\_address, Port))

server.listen(100)

list\_of\_clients = []

def clientthread(conn, addr):

conn.send("Welcome to this chatroom!") while True:

try:

message = conn.recv(2048)

if message:

print "<" + addr[0] + "> " + message

message\_to\_send = "<" + addr[0] + "> " + message

broadcast(message\_to\_send, conn)

else:

remove(conn)

except:

continue

def broadcast(message, connection):

for clients in list\_of\_clients:

if clients!=connection:

try:

clients.send(message)

except:

clients.close()

remove(clients)

def remove(connection):

if connection in list\_of\_clients:

list\_of\_clients.remove(connection)

while True:

conn, addr = server.accept()

list\_of\_clients.append(conn)

print addr[0] + " connected"

start\_new\_thread(clientthread,(conn,addr))

conn.close()

server.close()

**[C] Code for /client.py**

import socket

import select

import sys

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

if len(sys.argv) != 3:

print "Correct usage: script, IP address, port number"

exit()

IP\_address = str(sys.argv[1])

Port = int(sys.argv[2])

server.connect((IP\_address, Port))

while True:

sockets\_list = [sys.stdin, server]

read\_sockets,write\_socket, error\_socket = select.select(sockets\_list,[],[])

for socks in read\_sockets:

if socks == server:

message = socks.recv(2048)

print message

else:

message = sys.stdin.readline()

server.send(message)

sys.stdout.write("<You>")

sys.stdout.write(message)

sys.stdout.flush()

server.close()

**[D] Code for /GUI/SignInDialogGUI.py**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(525, 273)

self.label = QtWidgets.QLabel(Dialog)

self.label.setGeometry(QtCore.QRect(0, 5, 521, 41))

font = QtGui.QFont()

font.setPointSize(12)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.lineEdit = QtWidgets.QLineEdit(Dialog)

self.lineEdit.setGeometry(QtCore.QRect(210, 60, 201, 31))

self.lineEdit.setObjectName("lineEdit")

self.label\_2 = QtWidgets.QLabel(Dialog)

self.label\_2.setGeometry(QtCore.QRect(130, 60, 81, 31))

self.label\_2.setObjectName("label\_2")

self.label\_3 = QtWidgets.QLabel(Dialog)

self.label\_3.setGeometry(QtCore.QRect(130, 100, 81, 31))

self.label\_3.setObjectName("label\_3")

self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_2.setGeometry(QtCore.QRect(210, 100, 201, 31))

self.lineEdit\_2.setObjectName("lineEdit\_2")

self.pushButton = QtWidgets.QPushButton(Dialog)

self.pushButton.setGeometry(QtCore.QRect(210, 140, 93, 28))

self.pushButton.setObjectName("pushButton")

self.line = QtWidgets.QFrame(Dialog)

self.line.setGeometry(QtCore.QRect(100, 210, 331, 16))

self.line.setFrameShape(QtWidgets.QFrame.HLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.pushButton\_2 = QtWidgets.QPushButton(Dialog)

self.pushButton\_2.setGeometry(QtCore.QRect(210, 230, 93, 28))

self.pushButton\_2.setObjectName("pushButton\_2")

self.label\_4 = QtWidgets.QLabel(Dialog)

self.label\_4.setGeometry(QtCore.QRect(0, 170, 521, 41))

font = QtGui.QFont()

font.setPointSize(12)

self.label\_4.setFont(font)

self.label\_4.setAlignment(QtCore.Qt.AlignCenter)

self.label\_4.setObjectName("label\_4")

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

self.label.setText(\_translate("Dialog", "Sign-In"))

self.label\_2.setText(\_translate("Dialog", "Username"))

self.label\_3.setText(\_translate("Dialog", "Password"))

self.pushButton.setText(\_translate("Dialog", "Sign-In"))

self.pushButton\_2.setText(\_translate("Dialog", "Register"))

self.label\_4.setText(\_translate("Dialog", "OR"))

**[E] RegisterDialogGUI.py**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(525, 353)

self.label\_2 = QtWidgets.QLabel(Dialog)

self.label\_2.setGeometry(QtCore.QRect(130, 90, 81, 31))

self.label\_2.setObjectName("label\_2")

self.label\_3 = QtWidgets.QLabel(Dialog)

self.label\_3.setGeometry(QtCore.QRect(130, 230, 81, 31))

self.label\_3.setObjectName("label\_3")

self.label = QtWidgets.QLabel(Dialog)

self.label.setGeometry(QtCore.QRect(0, 5, 521, 41))

font = QtGui.QFont()

font.setPointSize(12)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_2.setGeometry(QtCore.QRect(210, 230, 201, 31))

self.lineEdit\_2.setText("")

self.lineEdit\_2.setObjectName("lineEdit\_2")

self.lineEdit = QtWidgets.QLineEdit(Dialog)

self.lineEdit.setGeometry(QtCore.QRect(210, 90, 201, 31))

self.lineEdit.setText("")

self.lineEdit.setObjectName("lineEdit")

self.pushButton = QtWidgets.QPushButton(Dialog)

self.pushButton.setGeometry(QtCore.QRect(210, 310, 93, 28))

self.pushButton.setObjectName("pushButton")

self.label\_4 = QtWidgets.QLabel(Dialog)

self.label\_4.setGeometry(QtCore.QRect(130, 50, 81, 31))

self.label\_4.setObjectName("label\_4")

self.lineEdit\_3 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_3.setGeometry(QtCore.QRect(210, 50, 201, 31))

self.lineEdit\_3.setText("")

self.lineEdit\_3.setObjectName("lineEdit\_3")

self.lineEdit\_4 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_4.setGeometry(QtCore.QRect(210, 190, 201, 31))

self.lineEdit\_4.setText("")

self.lineEdit\_4.setObjectName("lineEdit\_4")

self.lineEdit\_5 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_5.setGeometry(QtCore.QRect(210, 50, 201, 31))

self.lineEdit\_5.setText("")

self.lineEdit\_5.setObjectName("lineEdit\_5")

self.label\_5 = QtWidgets.QLabel(Dialog)

self.label\_5.setGeometry(QtCore.QRect(130, 190, 81, 31))

self.label\_5.setObjectName("label\_5")

self.line = QtWidgets.QFrame(Dialog)

self.line.setGeometry(QtCore.QRect(90, 170, 371, 20))

self.line.setFrameShape(QtWidgets.QFrame.HLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.label\_6 = QtWidgets.QLabel(Dialog)

self.label\_6.setGeometry(QtCore.QRect(130, 130, 81, 31))

self.label\_6.setObjectName("label\_6")

self.lineEdit\_6 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_6.setGeometry(QtCore.QRect(210, 130, 201, 31))

self.lineEdit\_6.setText("")

self.lineEdit\_6.setObjectName("lineEdit\_6")

self.label\_7 = QtWidgets.QLabel(Dialog)

self.label\_7.setGeometry(QtCore.QRect(130, 270, 81, 31))

self.label\_7.setObjectName("label\_7")

self.lineEdit\_7 = QtWidgets.QLineEdit(Dialog)

self.lineEdit\_7.setGeometry(QtCore.QRect(210, 270, 201, 31))

self.lineEdit\_7.setText("")

self.lineEdit\_7.setObjectName("lineEdit\_7")

self.pushButton\_2 = QtWidgets.QPushButton(Dialog)

self.pushButton\_2.setGeometry(QtCore.QRect(310, 310, 93, 28))

self.pushButton\_2.setObjectName("pushButton\_2")

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

self.label\_2.setText(\_translate("Dialog", "Last Name"))

self.label\_3.setText(\_translate("Dialog", "Password"))

self.label.setText(\_translate("Dialog", "Sign-In"))

self.pushButton.setText(\_translate("Dialog", "Register"))

self.label\_4.setText(\_translate("Dialog", "First Name"))

self.label\_5.setText(\_translate("Dialog", "Username"))

self.label\_6.setText(\_translate("Dialog", "Email"))

self.label\_7.setText(\_translate("Dialog", "Confirm Pass"))

self.pushButton\_2.setText(\_translate("Dialog", "Reset"))