МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 311 «Прикладные программные средства и математические методы»

**Лабораторная работа №7**

по дисциплине:

«Информационная безопасность»

на тему:

**«**RSA»

Работу выполнил:

студент гр. М3О-318Бк-19

Артиков Темур Улугбекович

Научный руководитель:

ассистент, Кос О.И.

Дата выдачи задания: 18.09.2021 .

Дата сдачи: 27.11.2021 .

.

Москва 2021 г.

Оглавление:

Глава 1: Введение…………………………………………………………………

Глава 2: Алгоритм для реализации………………………………………………

Глава 3: Код программы………………………………………………………….

Глава 4: Демонстрация работы………………………………………………......

Выводы………...…………………………………………………………………..

Список литературы……………………………………………………………….

**Глава 1: Введение**

**Цель работы:** программирование шифр RSA (шифрование, дешифрование)

Данная лабораторная работа была выполнена с помощью языка программирования Python в среде программирования

**Глава 2: Алгоритм для реализации**

Открытый ключ, который мы реализуем, основан на алгоритме RSA (Rivest - Shamir - Adleman), предложенном в 1977 году. Название алгоритма образовано первыми буквами фамилий его изобретателей: Рональд Ривест, Ади Шамир и Леонард Адлеман. В RSA используются большие простые числа, содержащие сотни цифр. В алгоритме открытого ключа генерируются два случайных простых числа, которые затем сложным образом комбинируются для создания открытого и закрытого ключей.

RSA (подобно другим криптосистемам с открытым ключом) позволяет не только шифровать послания, но и снабжать цифровой подписью файл или текст. Например, Алиса может зашифровать сообщение своим закрытым ключом, создав таким образом шифротекст, который можно дешифровать только с помощью открытого ключа Алисы. Этот шифротекст становится цифровой подписью к файлу. Фактически он даже не является секретным, так как любой человек в мире может прочитать его с помощью общедоступного открытого ключа Алисы. Суть в другом: шифруя сообщение с помощью своего закрытого ключа Алиса снабжает сообщение цифровой подписью способом, не допускающим ее подделку. Поскольку единственным обладателем закрытого ключа является Алиса, только она могла создать такой шифротекст, а значит, она уже не сможет утверждать, что Боб подделал или изменил сообщение!

Гарантия того, что подписанные сообщения впоследствии не сможет отрицать своего авторства, называется невозможность отказа (non-repudiation).

**Порядок генерирования открытых и закрытых ключей**

В криптосистемах с открытым ключом каждый ключ состоит из двух чисел. Открытым ключом служат числа n и е, закрытым - числа *n* и *d*. Эти числа создаются в три этапа:

1. Создайте два очень больших простых числа: *р* и *q*. Это должны быть два разных случайных числа. Перемножьте их для получения числа *n*.

2. Создайте случайное число *е*, взаимно простое с числом *(р - 1) (q - 1).*

3. Вычислите модульное обращение числа *е* — это будет число *d*.

Число *n* используется в обоих ключах. Число *d* должно храниться в тайне, поскольку оно позволяет дешифровывать сообщения. Теперь все готово для написания программы, которая будет генерировать эти ключи.

**Глава 3: Код программы**

**Исходный код программы makePublicPrivateKeys.py**:

import random, sys, os, primeNum, cryptomath

def main():

# Create a public/private keypair with 1024 bit keys:

print('Создание ключевых файлов...')

makeKeyFiles('al\_sweigart', 1024)

print('Сделаны ключевые файлы.')

def generateKey(keySize):

# Creates a public/private keys keySize bits in size.

p = 0

q = 0

# Step 1: Create two prime numbers, p and q. Calculate n = p \* q.

print('Генерация простых чисел p & q...')

while p == q:

p = primeNum.generateLargePrime(keySize)

q = primeNum.generateLargePrime(keySize)

n = p \* q

# Step 2: Create a number e that is relatively prime to (p-1)\*(q-1):

print('Генерация e, которая относительно проста для (p-1)\*(q-1)...')

while True:

# Keep trying random numbers for e until one is valid:

e = random.randrange(2 \*\* (keySize - 1), 2 \*\* (keySize))

if cryptomath.gcd(e, (p - 1) \* (q - 1)) == 1:

break

# Step 3: Calculate d, the mod inverse of e:

print('Вычисление d, которое является модулем, обратным e...')

d = cryptomath.findModInverse(e, (p - 1) \* (q - 1))

publicKey = (n, e)

privateKey = (n, d)

print('Public key:', publicKey)

print('Private key:', privateKey)

return (publicKey, privateKey)

def makeKeyFiles(name, keySize):

# Our safety check will prevent us from overwriting our old key files:

if os.path.exists('%s\_pubkey.txt' % (name)) or os.path.exists('%s\_privkey.txt' % (name)):

sys.exit('ПРЕДУПРЕЖДЕНИЕ: Файл %s\_pubkey.txt или %s\_privkey.txt уже существует! Используйте другое имя или удалите эти файлы и повторно запустите эту программу.' % (name, name))

publicKey, privateKey = generateKey(keySize)

print()

print('Открытый ключ представляет собой %s и %s-значный номер.' % (len(str(publicKey[0])), len(str(publicKey[1]))))

print('Запись открытого ключа в файл %s\_pubkey.txt ...' % (name))

fo = open('%s\_pubkey.txt' % (name), 'w')

fo.write('%s,%s,%s' % (keySize, publicKey[0], publicKey[1]))

fo.close()

print()

print('Закрытый ключ состоит из %s и %s-значного числа.' % (len(str(publicKey[0])), len(str(publicKey[1]))))

print('Запись закрытого ключа в файл%s\_privkey.txt ...' % (name))

fo = open('%s\_privkey.txt' % (name), 'w')

fo.write('%s,%s,%s' % (keySize, privateKey[0], privateKey[1]))

fo.close()

# If makePublicPrivateKeys.py is run (instead of imported as a module),

# call the main() function:

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

main()

**Исходный код программы rsaCipher.py**:

import sys

DEFAULT\_BLOCK\_SIZE = 128

BYTE\_SIZE = 256

def main():

filename = 'encrypted\_file.txt' # файл для записи и чтения

mode = input("Viberite mode [E]ncrypt | [D]ecrypt: ")

if mode == 'E':

message = input("Vvedite text: ")

pubKeyFilename = 'al\_sweigart\_pubkey.txt'

print('Ваш текст шифруется и записывается в %s...' % (filename))

encryptedText = encryptAndWriteToFile(filename, pubKeyFilename, message)

print('Шифрованный текст:')

print(encryptedText)

elif mode == 'D':

privKeyFilename = 'al\_sweigart\_privkey.txt'

print('Читаю из файла %s и продолжаю расшифрование...' % (filename))

decryptedText = readFromFileAndDecrypt(filename, privKeyFilename)

print('Расшифрованный текст:')

print(decryptedText)

def getBlocksFromText(message, blockSize=DEFAULT\_BLOCK\_SIZE):

messageBytes = message.encode('ascii') # convert the string to bytes

blockInts = []

for blockStart in range(0, len(messageBytes), blockSize):

blockInt = 0

for i in range(blockStart, min(blockStart + blockSize, len(messageBytes))):

blockInt += messageBytes[i] \* (BYTE\_SIZE \*\* (i % blockSize))

blockInts.append(blockInt)

return blockInts

def getTextFromBlocks(blockInts, messageLength, blockSize=DEFAULT\_BLOCK\_SIZE):

message = []

for blockInt in blockInts:

blockMessage = []

for i in range(blockSize - 1, -1, -1):

if len(message) + i < messageLength:

asciiNumber = blockInt // (BYTE\_SIZE \*\* i)

blockInt = blockInt % (BYTE\_SIZE \*\* i)

blockMessage.insert(0, chr(asciiNumber))

message.extend(blockMessage)

return ''.join(message)

def encryptMessage(message, key, blockSize=DEFAULT\_BLOCK\_SIZE):

encryptedBlocks = []

n, e = key

for block in getBlocksFromText(message, blockSize):

# ciphertext = plaintext ^ e mod n

encryptedBlocks.append(pow(block, e, n))

return encryptedBlocks

def decryptMessage(encryptedBlocks, messageLength, key, blockSize=DEFAULT\_BLOCK\_SIZE):

decryptedBlocks = []

n, d = key

for block in encryptedBlocks:

# plaintext = ciphertext ^ d mod n

decryptedBlocks.append(pow(block, d, n))

return getTextFromBlocks(decryptedBlocks, messageLength, blockSize)

def readKeyFile(keyFilename):

fo = open(keyFilename)

content = fo.read()

fo.close()

keySize, n, EorD = content.split(',')

return (int(keySize), int(n), int(EorD))

def encryptAndWriteToFile(messageFilename, keyFilename, message, blockSize=DEFAULT\_BLOCK\_SIZE):

keySize, n, e = readKeyFile(keyFilename)

if keySize < blockSize \* 8: # \* 8 to convert bytes to bits

sys.exit('ОШИБКА: Размер блока составляет %s бит, а размер ключа - %s бит. Шифр RSA требует, чтобы размер блока был равен или больше размера ключа. Либо уменьшите размер блока.' % (blockSize \* 8, keySize))

# Encrypt the message

encryptedBlocks = encryptMessage(message, (n, e), blockSize)

for i in range(len(encryptedBlocks)):

encryptedBlocks[i] = str(encryptedBlocks[i])

encryptedContent = ','.join(encryptedBlocks)

encryptedContent = '%s\_%s\_%s' % (len(message), blockSize, encryptedContent)

fo = open(messageFilename, 'w')

fo.write(encryptedContent)

fo.close()

return encryptedContent

def readFromFileAndDecrypt(messageFilename, keyFilename):

keySize, n, d = readKeyFile(keyFilename)

fo = open(messageFilename)

content = fo.read()

messageLength, blockSize, encryptedMessage = content.split('\_')

messageLength = int(messageLength)

blockSize = int(blockSize)

if keySize < blockSize \* 8: # \* 8 to convert bytes to bits

sys.exit('ОШИБКА: Размер блока составляет %s бит, а размер ключа - %s бит. Шифр RSA требует, чтобы размер блока был равен или больше размера ключа. Вы указали правильный файл ключа и зашифрованный файл?' % (blockSize \* 8, keySize))

encryptedBlocks = []

for block in encryptedMessage.split(','):

encryptedBlocks.append(int(block))

return decryptMessage(encryptedBlocks, messageLength, (n, d), blockSize)

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

main()

**Глава 4: Демонстрация работы**

1. Для начала работы с шифрованием требуется создать, так называемые, открытые и закрытые ключи. Запускаем программу makePublicPrivateKeys.py. В папке с программой автоматически создаются 2 новых файла: al\_sweigart\_privkey.txt и al\_sweigart\_pubkey.txt



Рисунок 1 – Работа программы makePublicPrivateKeys.py.

1. Запускаем программу rsaCipher.py и выбираем функцию шифрования Encrypt, введя значение “E”, как показано на рис. 2.

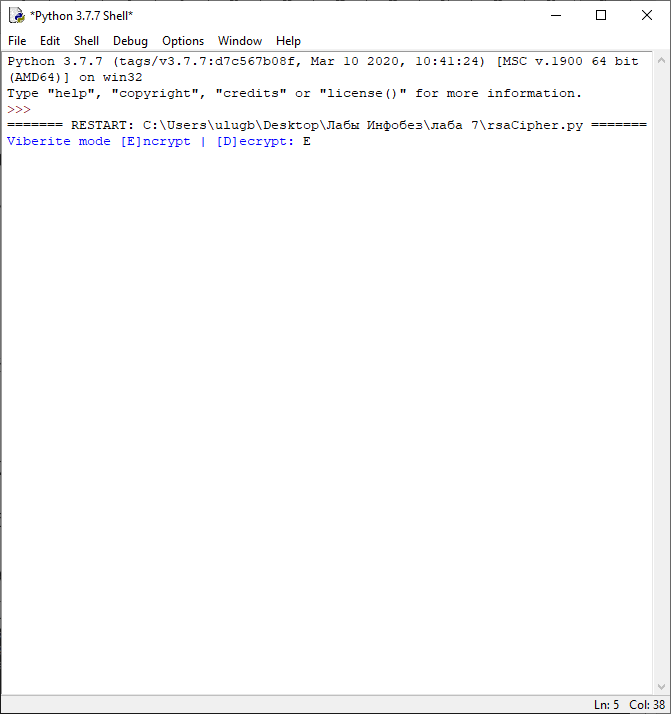


Рисунок 2 – Работа программы.

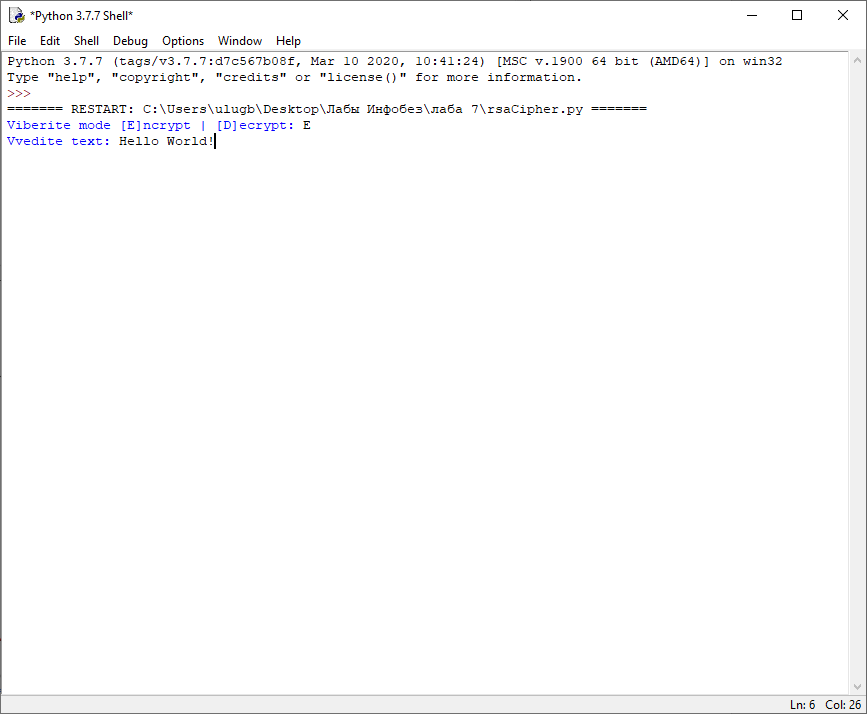
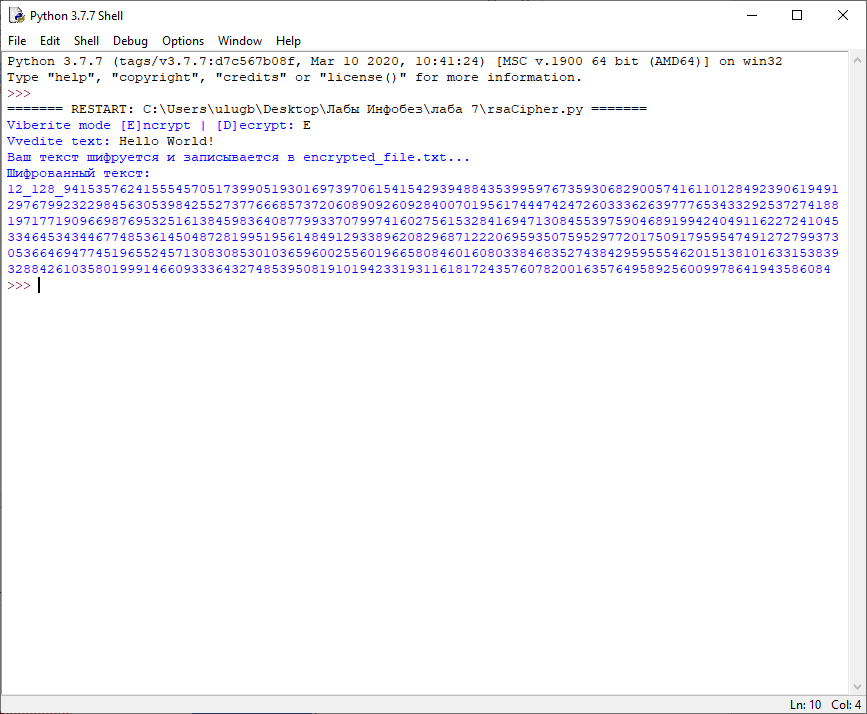
1. Ввод текста. Записанный текст: “Hello World!”  
    

Рис. 3 – Функция дешифрования.

1. На выходе программы, я получаю свой зашифрованный код. Текст шифра автоматически записывается в файле encrypted\_file.txt.

  
Рис. 4 – Шифрованный текст.

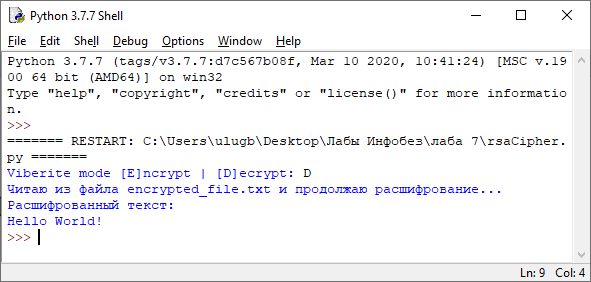
1. Дешифрование производится аналогичным способом. После выбора функции, вам не требуется вводить полученный текст вручную, т.е. программа сама найдёт и прочитает записанный файл encrypted\_file.txt и покажет изначальный текст, как показано на рис. 5.  
   

Рис. 5 – Дешифрование.

**Результаты тестирования программы:**

1. Ввод: «Hello World!». Вывод: результат написан в файле «encrypted\_file.txt»
2. Ввод: читается из файла «encrypted\_file.txt» Вывод: «Hello World!»

**Выводы**

Несмотря на то, что арифметика шифрования с открытым ключом везде одна и та же, вам не следует использовать рассмотренную здесь программу для защиты своих секретных файлов. Способы взлома подобных программ достаточно сложные, и тем не менее они существуют. Например, из-за того что случайные числа, генерируемые функцией random . randint (), не являются истинно случайными и могут быть предсказаны, злоумышленник способен выяснить, какие числа были выбраны в качестве простых чисел для вашего закрытого ключа.

Список литературы:

1. RSA wiki. <https://ru.wikipedia.org/wiki/RSA>
2. THE VIGENÈRE CIPHER. (Cracking Codes with Python)  
   <http://inventwithpython.com/hacking/chapter19.html>