Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №5**

по дисциплине: «Информационная безопасность и защита информации».

Выполнил:

студент 4 курса, гр. ИВТАПбд-41

Кондратьев Павел Сергеевич.

Проверил:

преподаватель кафедры ВТ

Мартынов Антон Иванович.

г. Ульяновск, 2019

**1) Задание:**

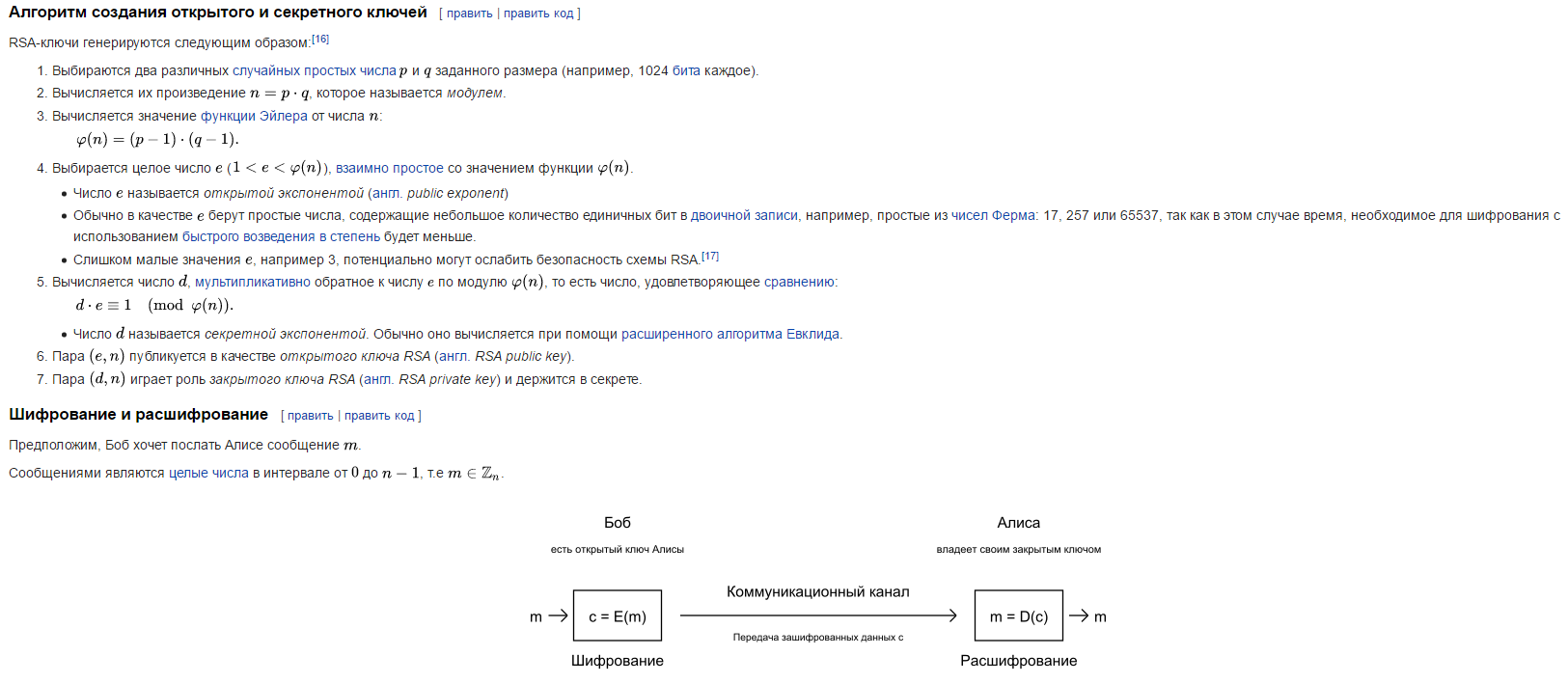
Реализовать приложение с графическим интерфейсом, позволяющее выполнять следующие действия:

1. Генерировать простые числа от заданной границы вправо
2. Создавать пару: Открытый ключ, Закрытый ключ
3. Шифровать текстовые и двоичные файлы Открытым ключом с помощью ассиметричного алгоритма, указанного в варианте задания
4. Дешифровать текстовые и двоичные файлы Закрытым ключом с помощью ассиметричного алгоритма, указанного в варианте задания
5. Сохранять зашифрованные/дешифрованные данные в файл
6. Загружать зашифрованные/дешифрованные данные из файла

**2) Краткие теоретические сведения:**

**Алгоритм RSA**

RSA–аббревиатура от сокращения фамилий 3-х ученых, разработавших эту систему –Рональда Райвеста, Ади Шамира и Леонарда Адельмана. Алгоритм был предложен в 1977 году. Он основывается на том, что нахождение больших простых чисел осуществляется сравнительно легко, но разложение на множители произведения двух таких чисел требует значительных вычислительных затрат.



**3) Порядок выполнения работы:**

1. Программа должна быть оформлена в виде удобной утилиты с интерактивным интерфейсом пользователя
2. Текст программы оформляется прилично (удобочитаемо, с описанием ВСЕХ функций, переменных и критических мест).
3. В процессе работы программа ОБЯЗАТЕЛЬНО выдает информацию о состоянии процесса генерации / тестирования (если процесс занимает длительное время)
4. Интерфейс программы может быть произвольным, но удобным и понятным (разрешается использование библиотек GUI)
5. Среда разработки и язык программирования могут быть произвольными

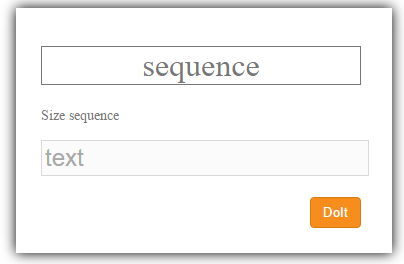


Рис1. Форма для задания начального текса

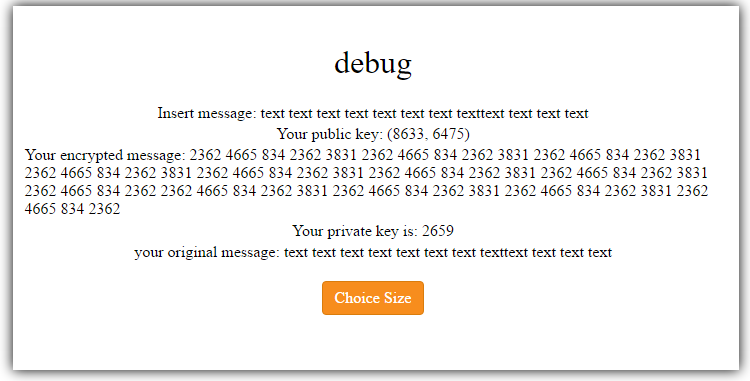


Рис4. Результат программы

**Вывод:**

В ходе выполнения данной лабораторной работы, мне удалось освоить базовую разработку графического интерфейса. Я научиться реализовывать методы ассиметричной криптографии: генерацию пары ключей (открытый ключ, закрытый ключ), шифрование сообщения с использованием открытого ключа, дешифрование с использованием закрытого ключа.

**Список литературы:**

* Руководство по PHP https://www.php.net/manual/ru/index.php (Дата обращения 2.12.19).
* Лабораторная работа Мартынов Антон Иванович - Информационная безопасность и защита информации.
* RSA https://ru.wikipedia.org/wiki/RSA (Дата обращения 8.12.19).

**Приложение 1**

import random

def \_try\_composite(a, d, n, s):

if pow(a, d, n) == 1:

return False

for i in range(s):

if pow(a, 2\*\*i \* d, n) == n-1:

return False

return True # n is definitely composite

\_known\_primes = [2, 3]

def isPrime(n, \_precision\_for\_huge\_n=16):

if n in \_known\_primes:

return True

if any((n % p) == 0 for p in \_known\_primes) or n in (0, 1):

return False

d, s = n - 1, 0

while not d % 2:

d, s = d >> 1, s + 1

# Returns exact according to http://primes.utm.edu/prove/prove2\_3.html

if n < 1373653:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3))

if n < 25326001:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5))

if n < 118670087467:

if n == 3215031751:

return False

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7))

if n < 2152302898747:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11))

if n < 3474749660383:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11, 13))

if n < 341550071728321:

return not any(\_try\_composite(a, d, n, s) for a in (2, 3, 5, 7, 11, 13, 17))

# otherwise

return not any(\_try\_composite(a, d, n, s)

for a in \_known\_primes[:\_precision\_for\_huge\_n])

def totient(number):

if(isPrime(number, 3)):

return number-1

else:

return False

# it isnt the best method to compute prime numbers

def prime(n): # check if the number is prime

if (n <= 1):

return False

if (n <= 3):

return True

if (n%2 == 0 or n%3 == 0):

return False

i = 5

while(i \* i <= n):

if (n%i == 0 or n%(i+2) == 0):

return False

i+=6

return True

def generate\_E(num):

def mdc(n1,n2):

rest = 1

while(n2 != 0):

rest = n1%n2

n1 = n2

n2 = rest

return n1

while True:

e = random.randrange(2,num)

if(mdc(num,e) == 1):

return e

def generate\_prime(): # generate the prime number - p e q

while True: # 2\*\*2048 is the RSA standart keys

x=random.randrange(1,100) # define the range of the primes

if(isPrime(x, 3)==True):

return x

def mod(a,b): # mod function

if(a<b):

return a

else:

c=a%b

return c

def mod(a,b): # mod function

if(a<b):

return a

else:

c=a%b

return c

def cipher(words,e,n): # get the words and compute the cipher

tam = len(words)

i = 0

lista = []

while(i < tam):

letter = words[i]

k = ord(letter)

k = k\*\*e

d = mod(k,n)

lista.append(d)

i += 1

return lista

def descifra(cifra,n,d):

lista = []

i = 0

tamanho = len(cifra)

# texto=cifra ^ d mod n

while i < tamanho:

result = cifra[i]\*\*d

texto = mod(result,n)

letra = chr(texto)

lista.append(letra)

i += 1

return lista

def calculate\_private\_key(toti,e):

d = 0

while(mod(d\*e,toti)!=1):

d += 1

return d

## MAIN

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

# text = input("Insert message: ")

f = open('text.txt')

text = f.read()

text\_input = "Insert message: " + text + ""

print(text\_input + '/')

p = generate\_prime() # generates random P

q = generate\_prime() # generates random Q

n = p\*q # compute N

y = totient(p) # compute the totient of P

x = totient(q) # compute the totient of Q

totient\_de\_N = x\*y # compute the totient of N

e = generate\_E(totient\_de\_N) # generate E

public\_key = (n, e)

print('Your public key: ', str(public\_key) + '/')

text\_cipher = cipher(text,e,n)

print('Your encrypted message: ', end='')

for code in text\_cipher:

print(code, end=' ')

print('/')

d = calculate\_private\_key(totient\_de\_N,e)

print('Your private key is:', str(d) + '/')

original\_text = descifra(text\_cipher,n,d)

print('your original message: ', end='')

for word in original\_text:

print(word, end='')

print('/')