Лабораторная работа №1

def ChastotnyAnaliz(text):

Frequencies = [0]\*(ord(max(text))+1)

for symbol in text:

Frequencies[ord(symbol)] += 1

return Frequencies

from sys import \*

#=====зашифровали=======================================

alpha = ' abcdefghijklmnopqrstuvwxyz'

print('enter the key:')

n = int(input())

s = open ('text.txt')

res = open ('cyphered.txt', 'w')

for c in s.read():

res.write(alpha[(alpha.find(c) + n) % len(alpha)])

res.close()

s.close()

#=====вывели частоты в книге=============================

cyp = open ('book.txt')

text\_book = cyp.read()

bookF = ChastotnyAnaliz(text\_book)

for ord\_symbol in range(len(bookF)):

if bookF[ord\_symbol] != 0:

print(chr(ord\_symbol), bookF[ord\_symbol]/len(text\_book))

cyp.close()

print('---------------------')

#=====вывели частоты в зашифрованном тексте=======================

cyp = open ('cyphered.txt')

text\_c = cyp.read()

cypheredF = ChastotnyAnaliz(text\_c)

for ord\_symbol in range(len(cypheredF)):

if cypheredF[ord\_symbol] != 0:

print(chr(ord\_symbol), cypheredF[ord\_symbol]/len(text\_c))

cyp.close()

print('-----------------------------------------------------')

#=====сравнили и нашли соответствие частот и ключ=======================

key = [0]\*300

for i in range(len(bookF)):

for j in range(len(cypheredF)):

if bookF[i]!=0:

if cypheredF[j]!=0:

if abs(bookF[i]/len(text\_book)-cypheredF[j]/len(text\_c))< 0.001:

print('{0:22} {1} - {2} {3}'.format(bookF[i]/len(text\_book), chr(i), chr(j) , cypheredF[j]/len(text\_c)))

key[(j - i)%27] += 1

print ('key =', key.index(max(key)))

#=====расшифровали=======================

s = open ('cyphered.txt')

res = open ('decyphered.txt', 'w')

for c in s.read():

res.write(alpha[(alpha.find(c) - key.index(max(key))) % len(alpha)])

res.close()

s.close()

Лабораторная работа №2

import string

import random

def random\_string(n):

a = string.ascii\_letters

return ''.join([random.choice(a) for i in range(n)])

def xor(message, key):

return bytes(a^b for a, b in zip(message, key))

print('введите слово:')

word = input()

print('сгенерированный ключ той же длины:')

key = random\_string(len(word))

print(key)

res = xor(word.encode('utf-8'), key.encode('utf-8')).decode('utf-8')

for byte1, byte2, byte3 in zip(word.encode('utf-8'), key.encode('utf-8'), res.encode('utf-8')):

print("{0:12b}".format(byte1))

print("xor {0:08b}".format(byte2))

print('-----------')

print("{0:12b}".format(byte3))

print(' ')

print ('результат шифрования:', res)

deres = xor(res.encode('utf-8'),key.encode('utf-8')).decode('utf-8')

for byte1, byte2, byte3 in zip(res.encode('utf-8'), key.encode('utf-8'), deres.encode('utf-8')):

print("{0:12b}".format(byte1))

print("xor {0:08b}".format(byte2))

print('-----------')

print("{0:12b}".format(byte3))

print(' ')

print ('результат дешифрования:', deres)

Лабораторная работа №3

from random import getrandbits

from random import randint

from fractions import gcd

import sys

def is\_prime\_calc(num):

return all(num % i for i in range(2, num))

def is\_prime(num):

return is\_prime\_calc(num)

def get\_random\_prime():

while True:

n = getrandbits(12) + 3;

if is\_prime(n):

return n

def primitive\_root(modulo):

required\_set = set(num for num in range (1, modulo) if gcd(num, modulo) == 1)

for g in range(1, modulo):

actual\_set = set(pow(g, powers) % modulo for powers in range (1, modulo))

if required\_set == actual\_set:

return g

class human:

p = 0

g = 0

secret\_num = 0

pulic\_key = 0

message = 0

secret\_key = 0

Alice = human()

Bob = human()

print(Alice)

print ('Обоим абонентам известны некоторые два числа g и p, которые не являются секретными')

p = get\_random\_prime()

g = primitive\_root(p)

print('p = %d' %p)

print('g = %d' %g)

Alice.p = p

Alice.g = g

Bob.p = p

Bob.g = g

print ('Для того, чтобы создать неизвестный более никому секретный ключ, оба абонента генерируют большие случайные числа: Алиса — a, Боб — b')

Alice.secret\_num = randint(1, 9999)

Bob.secret\_num = randint(1, 9999)

print('a = %d' %Alice.secret\_num)

print('b = %d' %Bob.secret\_num)

print('Затем Алиса вычисляет значение A, и пересылает его Бобу, а Боб вычисляет B и передаёт Алисе')

Alice.public\_key = (g\*\*Alice.secret\_num) % p

Bob.public\_key = (g\*\*Bob.secret\_num) % p

Alice.message = Bob.public\_key

Bob.message = Alice.public\_key

print('A = %d' %Alice.public\_key)

print('B = %d' %Bob.public\_key)

print('На втором этапе Алиса на основе имеющегося у неё a и полученного по сети B вычисляет значение K:')

Alice.secret\_key = (Bob.public\_key\*\*Alice.secret\_num) % p

print(Alice.secret\_key)

print('Боб на основе имеющегося у него b и полученного по сети A вычисляет значение K:')

Bob.secret\_key = (Alice.public\_key\*\*Bob.secret\_num) % p

print(Bob.secret\_key)

print('Как нетрудно видеть, у Алисы и Боба получилось одно и то же число. \

Его они и могут использовать в качестве секретного ключа')

print('Таким образом,\nАлиса знает:')

print(' открытое простое число p = {0}\n первообразный корень по модулю р g = {1}\n секретный ключ a={2}\n открытый ключ A={3}\n открытый ключ B={4}\n секретный ключ K={5}'\

.format(Alice.p, Alice.g, Alice.secret\_num, Alice.public\_key, Alice.message, Alice.secret\_key))

print('Боб знает:')

print(' открытое простое число p = {0}\n первообразный корень по модулю р g = {1}\n секретный ключ b={2}\n открытый ключ B={3}\n открытый ключ A={4}\n секретный ключ K={5}'\

.format(Bob.p, Alice.g, Bob.secret\_num, Bob.public\_key, Bob.message, Bob.secret\_key))

Лабораторная работа №4

from random import getrandbits

from random import randint

from fractions import gcd

def is\_prime(num):

return all(num % i for i in range(2, num))

def random\_prime():

while True:

n = getrandbits(12) + 3;

if is\_prime(n):

return n

def egcd(a, b):

if a == 0:

return (b, 0, 1)

g, y, x = egcd(b%a,a)

return (g, x - (b//a) \* y, y)

def modinv(a, m):

g, x, y = egcd(a, m)

if g != 1:

raise Exception('No modular inverse')

return x%m

def get\_e(e, phi):

while gcd(e, phi) != 1:

e = randint(1, phi)

return e

p = random\_prime()

q = random\_prime()

n = p\*q

phi = (p-1)\*(q-1)

e = get\_e(0, phi)

d = modinv(e, phi)

print('2 случайных простых числа')

print('P =', p)

print('Q =', q)

print('Модуль N =', n)

print('Phi(N) =', phi)

print('Открытая экспонента E =', e)

print('Мультипликативно обратное к Е число D =', d)

print('(E\*D)%Phi =', (e\*d)%phi)

print('Открытый ключ RSA: [e {0}, n {1}]'.format(e, n))

print('Закрытый ключ RSA: [ф {0}, d {1}, p {2}, q {3}]'.format(phi, d, p, q))

print('Сообщение:')

message = int(input())

print('Зашифрованное с помощью открытого ключа сообщение:')

c = pow(message, e, n)

print(c)

print('Расшифрованное с помощью закрытого ключа сообщение:')

m = pow(c, d, n)

print(m)

Лабораторная работа №6

from random import getrandbits

from random import randint

from fractions import gcd

import random, string

def randomword(length):

letters = string.ascii\_lowercase

return ''.join(random.choice(letters) for i in range(length))

def is\_prime(num):

return all(num % i for i in range(2, num))

def safe\_prime():

while True:

q = getrandbits(12) + 3;

if is\_prime(q):

n = 2\*q+1

if is\_prime(n):

print ('q =', q)

return n

def get\_g(g, n):

while gcd(g, n) != 1:

g = randint(1, n)

return g

class switch(object):

def \_\_init\_\_(self, value):

self.value = value # значение, которое будем искать

self.fall = False # для пустых case блоков

def \_\_iter\_\_(self): # для использования в цикле for

""" Возвращает один раз метод match и завершается """

yield self.match

raise StopIteration

def match(self, \*args):

""" Указывает, нужно ли заходить в тестовый вариант """

if self.fall or not args:

# пустой список аргументов означает последний блок case

# fall означает, что ранее сработало условие и нужно заходить

# в каждый case до первого break

return True

elif self.value in args:

self.fall = True

return True

return False

class user:

I = ''

s = ''

x = 0

v = 0

A = 0

B = 0

u = 0

K = 0

M = 0

R = 0

class server:

I = ''

s = ''

x = 0

v = 0

A = 0

B = 0

u = 0

K = 0

M = 0

R = 0

# init

client = []

s\_data = []

i = 0

g = 0

ch = 1

# safe field

N = safe\_prime()

g = get\_g(g, N)

k = 3

print('безопасное простое N = 2\*q +1 = ', N)

print('генератор по модулю N g =', g)

print('параметр-множитель k =', k)

while (ch!=0):

print('0 - выход\

1 - регистрация \

2 - аутентификация:')

ch = int(input())

for case in switch(ch):

if case(0):

break

if case(1):

client = user()

s\_data = server()

print('Регистрация')

print('Введите имя:')

client.I = input()

print('Введите пароль:')

p = input()

client.s = randomword(3)

client.x = abs(hash(client.s + p)%N) #secret key

client.v = g\*\*(client.x) % N #верификатор пароля на стороне сервера

s\_data.I = client.I

s\_data.s = client.s

s\_data.v = client.v

break

if case(2):

print('Аутентификация')

print('Введите имя:')

name = input()

print('Введите пароль:')

pwd = input()

if name == s\_data.I:

a = randint(1, 10)

client.A = g\*\*a % N

if client.A != 0:

s\_data.A = client.A

else:

print ('A=0')

b = randint(1, 10)

s\_data.B = s\_data.v\*k + g\*\*b % N

if s\_data.B != 0:

client.B = s\_data.B

else:

print ('B=0')

# вычисление скремблера

client.u = abs(hash(str(client.A)+str(client.B))%N)

s\_data.u = abs(hash(str(s\_data.A)+str(s\_data.B))%N)

x = abs(hash(client.s + pwd)%N)

CSession = ((client.B - k \* (g\*\*x % N))\*\*(a + client.u \* x)) % N

client.K = abs(hash(str(CSession))%N)

SSession = ((s\_data.A \* (s\_data.v\*\*s\_data.u % N))\*\*b) % N

s\_data.K = abs(hash(str(SSession))%N)

#второй раунд обмена

client.M = abs(hash(str(abs(hash(N)^hash(g))) + str(abs(hash(client.I))) + client.s + str(client.A) + str(client.B) + str(client.K)))

s\_data.M = abs(hash(str(abs(hash(N)^hash(g))) + str(abs(hash(s\_data.I))) + s\_data.s + str(s\_data.A) + str(s\_data.B) + str(s\_data.K)))

if s\_data.M == client.M:

client.R = abs(hash(str(client.A) + str(client.M) + str(client.K)))

s\_data.R = abs(hash(str(s\_data.A) + str(s\_data.M) + str(s\_data.K)))

print('Вход выполнен')

ch = 0

else:

print('Неверный пароль')

else:

print('Неверное имя')

break