# Аннотация

**Среда программирования:** Visual Studio Code

**Язык программирования:** Python 3

**Процедуры для запуска программы:** $ python3 <имя\_файла>.py

**Пословица-тест:** Красивыми словами пастернак не помаслишь

**Текст для проверки работы:** Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

**Интерфейс:** #в разработке#

**ГЕНЕРАЦИЯ ЦИФРОВОЙ ПОДПИСИ**



## 

* **Elgamal**

**Код программы:**

from math import gcd

import random

#инициализация алфавита

alphavit = {'а':0, 'б':1, 'в':2, 'г':3, 'д':4,

                  'е':5, 'ж':6, 'з':7, 'и':8, 'й':9,

                  'к':10, 'л':11, 'м':12, 'н':13, 'о':14,

                  'п':15, 'р':16, 'с':17, 'т':18, 'у':19,

                  'ф':20, 'х':21, 'ц':22, 'ч':23, 'ш':24,

                  'щ':25, 'ъ':26, 'ы':27, 'ь':28, 'э':29,

                  'ю':30, 'я':31, ' ':32, ",":33, ".":34

                  }

#проверка на простое число

def IsPrime(n):

    d = 2

    while n % d != 0:

        d += 1

    return d == n

#расширенный алгоритм Евклида или (e\*\*-1) mod fe

def modInverse(e,el):

    e = e % el

    for x in range(1,el):

        if ((e \* x) % el == 1):

            return x

    return 1

#выбор простого целого P, выбор целого числа G,G<P

def is\_prime(num, test\_count):

    if num == 1:

        return False

    if test\_count >= num:

        test\_count = num - 1

    for x in range(test\_count):

        val = random.randint(1, num - 1)

        if pow(val, num-1, num) != 1:

            return False

    return True

def gen\_prime(n):

    found\_prime = False

    while not found\_prime:

        p = random.randint(2\*\*(n-1), 2\*\*n)

        if is\_prime(p, 1000):

            return p

p = gen\_prime(10)

print("P =",p)

print()

g = random.randint(2,p-1)

print("G =",g)

print()

#отправитель выбирает случайное целое число X,1<x<(p-1)

x = random.randint(2,p-2)

y = (g\*\*x)%p

print("Открытый ключ(Y)={}, Секретный ключ(X)={}".format(y,x))

print()

#хэшируем сообщение

msg = input("Введите сообщение:")

msg\_list = list(msg)

alpha\_code\_msg = list()

for i in range(len(msg\_list)):

    alpha\_code\_msg.append(int(alphavit.get(msg\_list[i])))

print("Длина исходного сообщения {} символов".format(len(alpha\_code\_msg)))

print()

def hash\_value(mod,alpha\_code):

    i = 0

    hashing\_value = 1

    while i < len(alpha\_code\_msg):

        hashing\_value = (((hashing\_value-1) + int(alpha\_code\_msg[i]))\*\*2) % mod

        i += 1

    return hashing\_value

hash\_code\_msg = hash\_value(p, alpha\_code\_msg)

print("Хэш сообщения:= {}".format(hash\_code\_msg))

print()

#генерация случайное целое число K

k = 1

while True:

    k = random.randint(1,p-2)

    if gcd(k,p-1) == 1:

        print("K =",k)

        break

#отправитель вычисляет число целое число а

a = (g\*\*k)%p

#вычисляем b

b = modInverse(k,p-1) \* ((hash\_code\_msg - (x \* a))%(p-1))

#b = modInverse((int(hash\_code\_msg) - int(x)\*int(a)),p-1)

print("Значение подписи:S={},{}".format(a,b))

print()

#првоерка подписи (передвём m, a,b)

check\_hash\_value = hash\_value(p, alpha\_code\_msg)

a\_1 = ((y\*\*a) \* (a\*\*b)) % p

print("A1={}".format(a\_1))

print()

a\_2 = (g\*\*check\_hash\_value)%p

print("A2={}".format(a\_2))

print()

if a\_1 == a\_2:

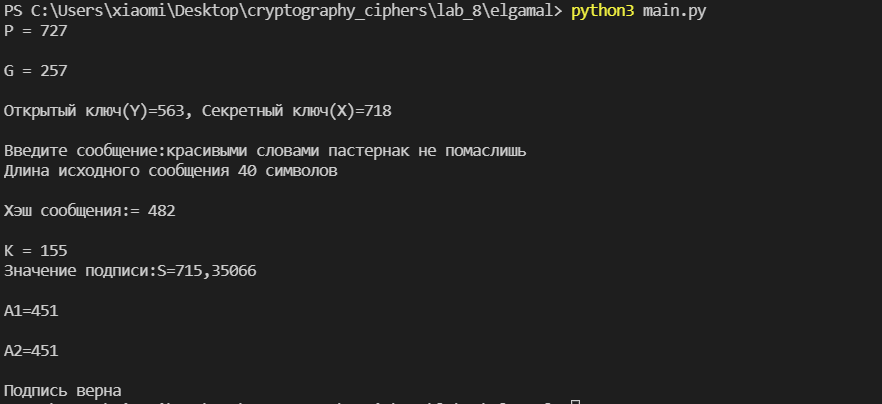
    print("Подпись верна")

else:

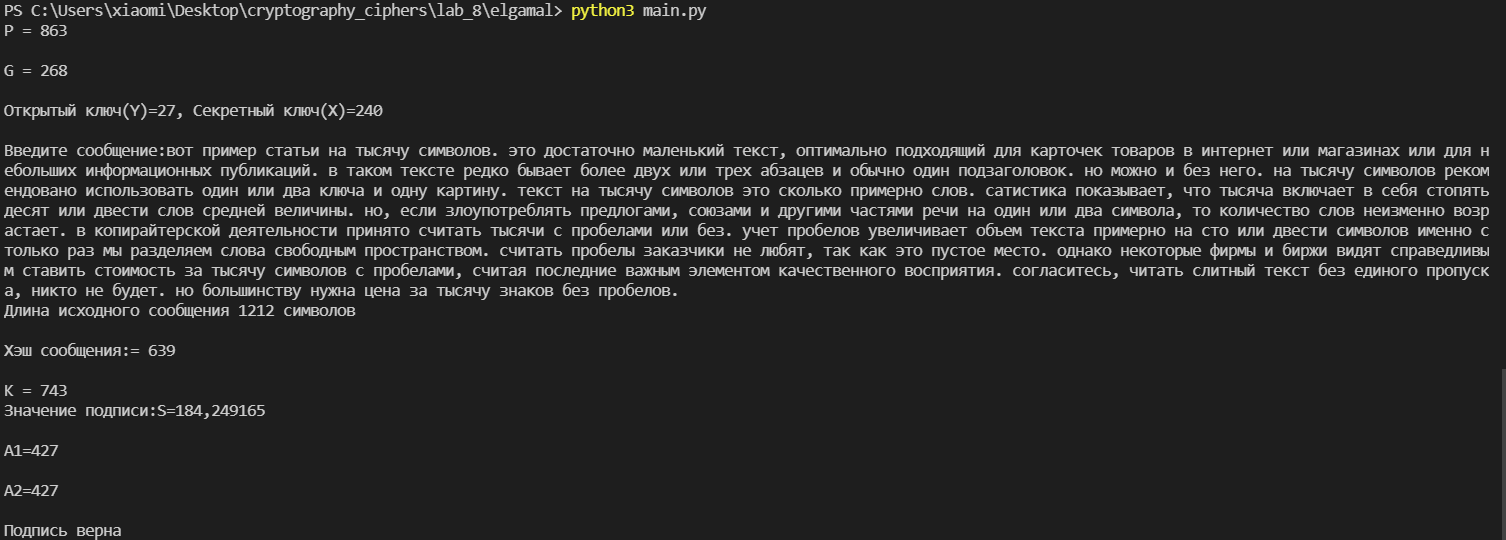
    print("Подпись неверна")

**Тестирование:**

**Фраза по варианту**



**Текст на 1000 символов**

****

* **RSA**

**Программа**

from math import gcd

#инициализация алфавита

alphabet\_lower = {'а':0, 'б':1, 'в':2, 'г':3, 'д':4,

                  'е':5, 'ж':6, 'з':7, 'и':8, 'й':9,

                  'к':10, 'л':11, 'м':12, 'н':13, 'о':14,

                  'п':15, 'р':16, 'с':17, 'т':18, 'у':19,

                  'ф':20, 'х':21, 'ц':22, 'ч':23, 'ш':24,

                  'щ':25, 'ъ':26, 'ы':27, 'ь':28, 'э':29,

                  'ю':30, 'я':31, ' ':32, ",":33, ".":34

                  }

#проверка на простое число

def IsPrime(n):

    d = 2

    while n % d != 0:

        d += 1

    return d == n

#расширенный алгоритм Евклида или (e\*\*-1) mod fe

def modInverse(e,el):

    e = e % el

    for x in range(1,el):

        if ((e \* x) % el == 1):

            return x

    return 1

#инициализация p,q,e,n

p = int(input("Введите p: "))

print(IsPrime(p))

q = int(input("Введите q: "))

print(IsPrime(q))

n = p \* q

print("N =",n)

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

print("El =",el)

e = 257

print("E =",e)

if gcd(e,el) == 1:

    print(gcd(e,el),"E подходит")

else:

    print(gcd(e,el),"False")

#нахождение секретной экспоненты D

d = modInverse(e,el)

print("D =",d)

print("Открытый ключ e={} n={}".format(e,n))

print("Секретный ключ d={} n={}".format(d,n))

#хэширование сообщения

msg = input("Введите сообщение:")

msg\_list = list(msg)

alpha\_code\_msg = list()

for i in range(len(msg\_list)):

    alpha\_code\_msg.append(int(alphabet\_lower.get(msg\_list[i])))

print("Длина исходного сообщения {} символов".format(len(alpha\_code\_msg)))

def hash\_value(n,alpha\_code):

    i = 0

    hashing\_value = 1

    while i < len(alpha\_code\_msg):

        hashing\_value = (((hashing\_value-1) + int(alpha\_code\_msg[i]))\*\*2) % n

        i += 1

    return hashing\_value

hash\_code\_msg = hash\_value(n, alpha\_code\_msg)

print("Хэш сообщения", hash\_code\_msg)

#подпись сообщения s=Sa(m) = m^d mod n

def signature\_msg(hash\_code,n,d):

    sign = (hash\_code\*\*d)%n

    return sign

sign\_msg = signature\_msg(hash\_code\_msg,n,d)

print("Значение подписи: {}".format(sign\_msg))

#передаём пару m,s

def check\_signature(sign\_msg, n,e):

    check = (sign\_msg\*\*e) % n

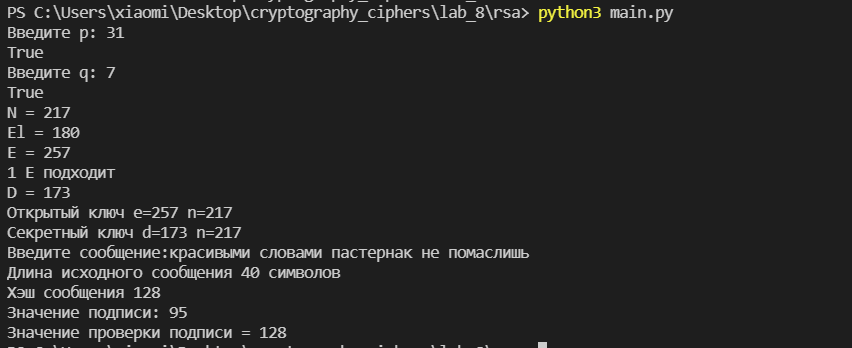
    return check

check\_sign = check\_signature(sign\_msg,n,e)

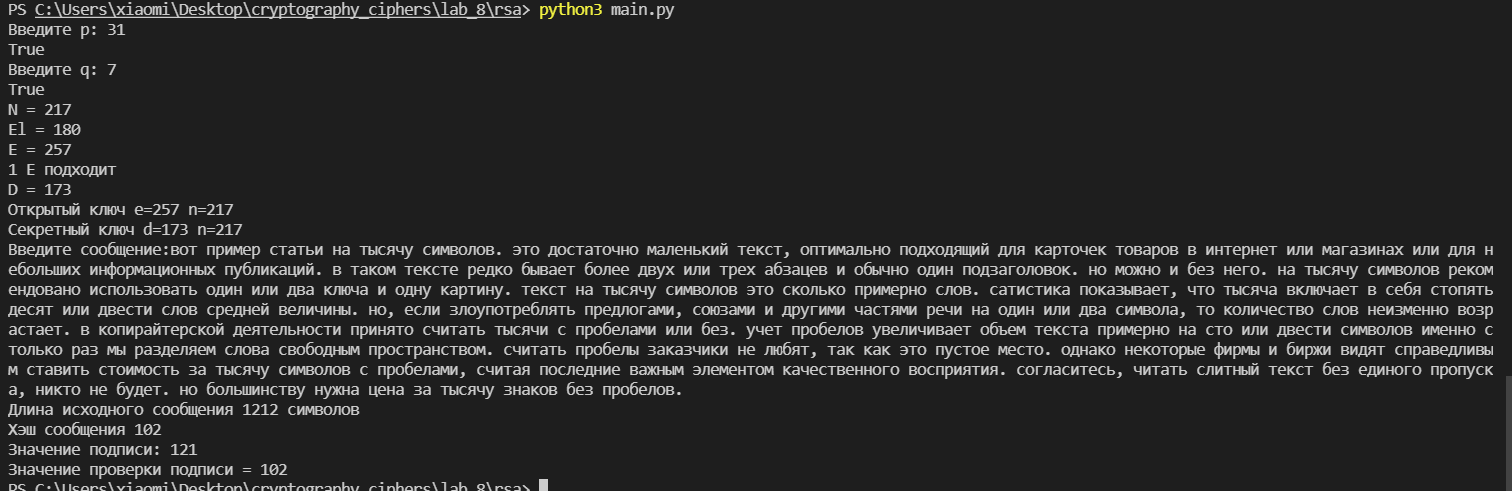
print("Значение проверки подписи = {}".format(check\_sign))

**Тестирование**

**Фраза по варианту**

****

**Проверка текста на 1000 символов**

****