Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Программирование криптографических алгоритмов

Блок Н: АЛГОРТИМЫ ЦИФРОВЫХ ПОДПИСЕЙ

Выполнила студентка 3 курса группы 171-341

Решетникова Дарья

Москва 2020 г.

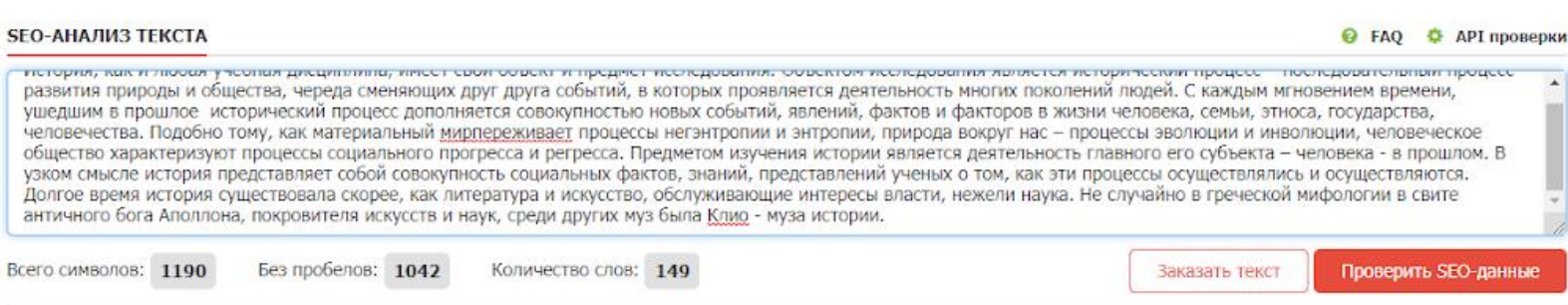
Аннотация

Язык : Python

Программа: Visual Studio 2017

Пословица: Плод никогда не падает далеко от дерева.

Текст: История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовательный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени, ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. Подобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют процессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представляет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и наук, среди других муз была Клио - муза истории.



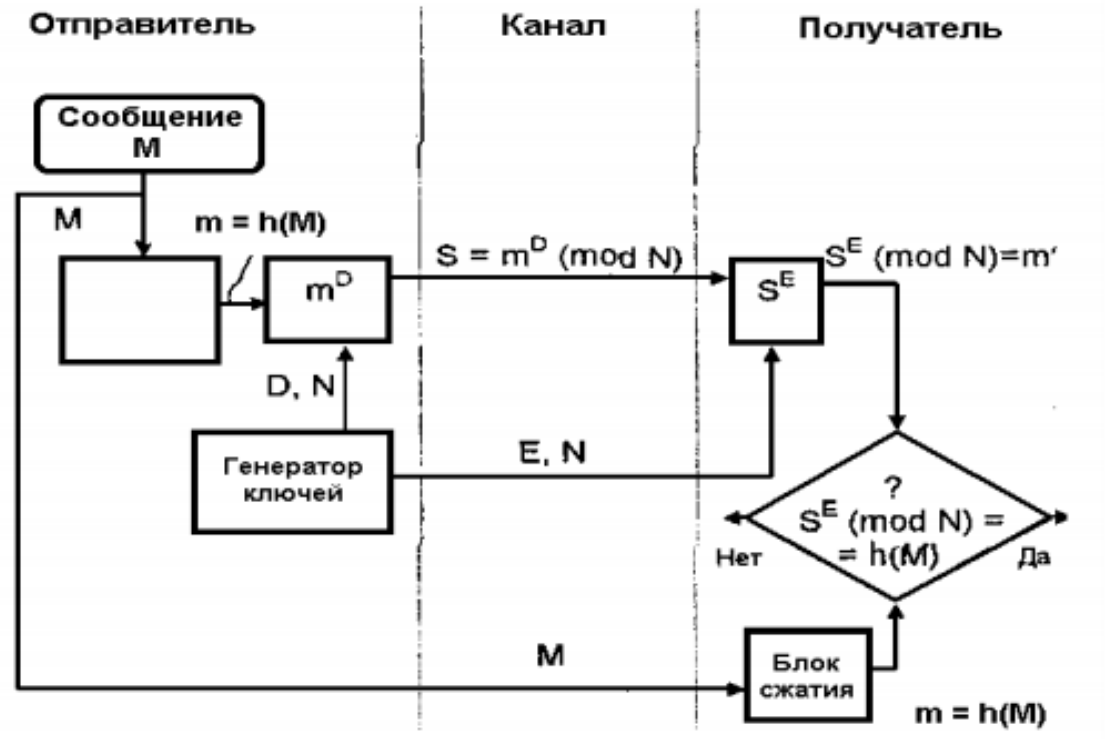
RSA.

1. Описание шифра.

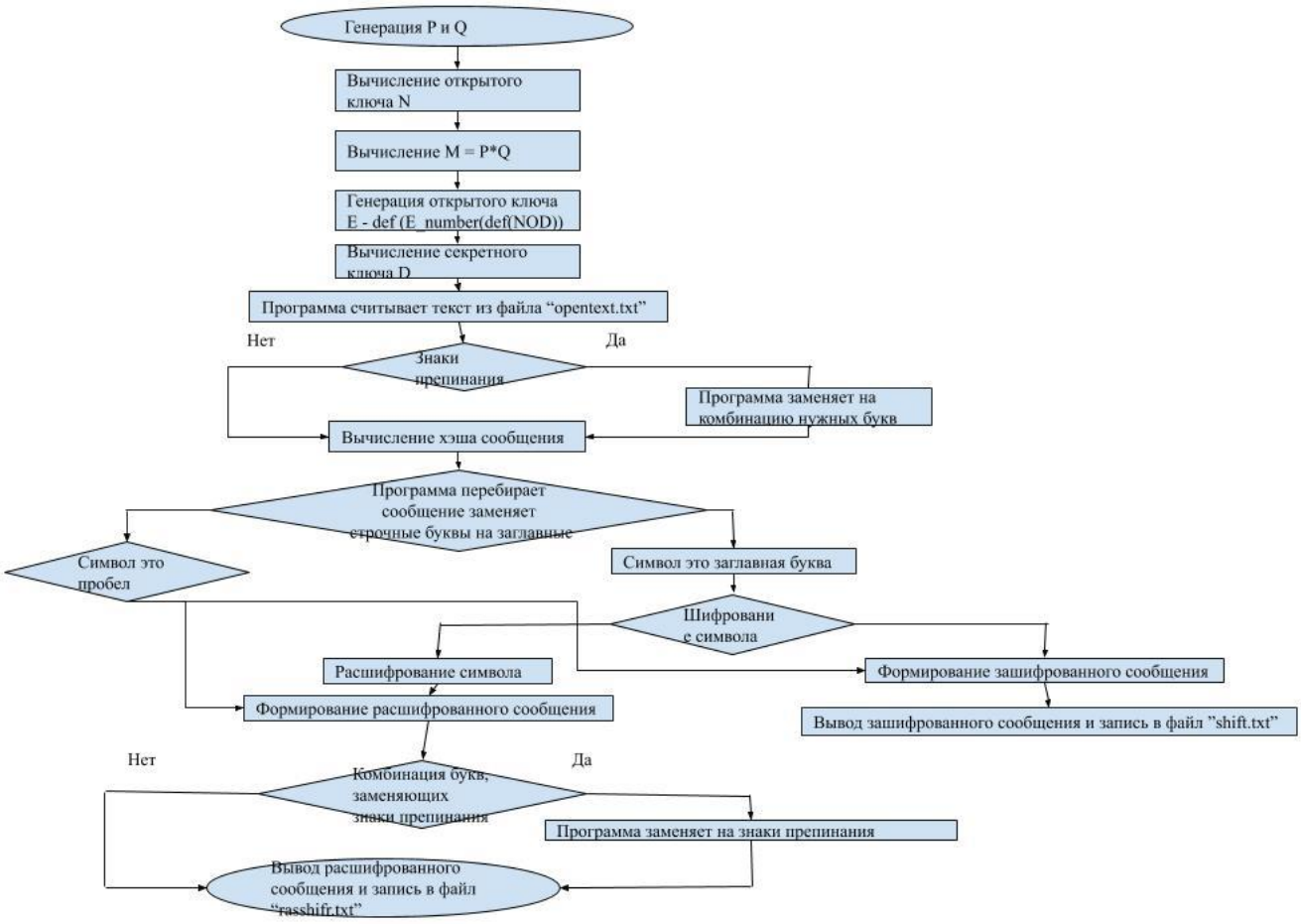
RSA - первый алгоритм цифровой подписи, который был разработан в 1977 году в Массачусетском технологическом институт и назван по первым буквам фамилий ее разработчиков (Ronald Rivest, Adi Shamir и Leonard Adleman). RSA основывается на сложности разложения большого числа n на простые множители.

2. Алгоритм шифра.

1. **Берутся два очень больших простых числа P и Q и находятся произведение простых чисел N=P×Q и функция Эйлера от этого произведения M=(P-1)×(Q-1).**
2. **Выбирается случайное целое число D, взаимно простое с M, и вычисляется**
   $$E=(1 \text{ MOD } M)/D.$$
3. **Потом D и N публикуются как открытый ключ, E сохраняется в тайне.**
4. **Если S — сообщение, длина которого, определяемая по значению выражаемого им целого числа, должна быть в интервале (1,N), то оно превращается в шифровку возведением в степень D по модулю N и отправляется получателю $S'=S^D \text{ MOD } N$.**
5. **Получатель сообщения расшифровывает его, возведя в степень E (число E ему уже известно) по модулю N, так как**
   $$S=(S'^E \text{ MOD } N)=S(^{DE}) \text{ MOD } N.$$



3. Блок-схема программы



4. Код программы

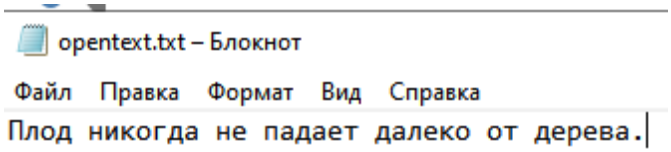
```
import math
from random import randrange
import re
print("Вычисления")
# генерация P и Q - секретные ключи - простые большие числа
def rand_prime():
    while True:
        p = randrange(9, 99,2)
        if all(p % n != 0 for n in range(3, int((p ** 0.5) + 1), 2)):
            return p
#P = 3
#Q = 11
P = rand_prime()
Q = rand_prime()
print("Первое случайное число: ", P)
print("Второе случайное число: ", Q)
# находим N - открытый ключ - произведение простых чисел
N=P*Q
print("Произведение чисел N = P*Q: ", N)
# находим M - функция Эйлера
M=(P-1)*(Q-1)
print("Функция Эйлера (M): ", M)
# находим E - открытый ключ - случайное число, взаимно простое с M
def NOD(a, b):
    while a != 0 and b != 0:
        if a > b:
            a %= b
        else:
            b %= a
    return a+b
def E_number(M):
```

```
    E = randrange(11, 99, 2)
    while not NOD(E,M) == 1:
        E = randrange(11, 99, 2)
    return E
#E = 7
E = E_number(M)
print("Число E: ", E)
# находим D - секретный ключ - вычисляется
D = 0
while not E * D % M == 1:
    D += 1
print("Число D: ", D, "\n")
print("Генерация ключей")
print('Открытые ключи D и N:', D,"и", N)
print('Закрытый ключ E: ', E)
# в этот файл нужно записать исходный текст
f = open(r"opentext.txt", "rt", encoding='utf-8')
text = f.read()
print("Исходное сообщение:", text)
text = text.upper()
text1 = text.upper()
text = text.replace('.', ' ТЧК') # Если в сообщении попадется точка, она заменется на тчк
text = text.replace(',', ' ЗПТ') # Если в сообщении попадется запятая, она заменется на
зпт
text = text.replace('-', 'ТИРЕ') # Если в сообщении попадется - (тире), оно заменется на
тире
print("Преобразованное исходное сообщение:", text)
blst
={'А','Б','В','Г','Д','Е','Ж','З','И','Й','К','Л','М','Н','О','П','Р','С','Т','У','Ф','Х'
,'Ц','Ч','Ш','Щ','Ъ','Ы','Ь','Э','Ю','Я'} # задаем алфавит заглавных букв
# Вычисляем хэш исходного сообщения
dlina = 0
probel = ""
for symbol in text:
    if symbol == " ":
        probel += " "
    elif symbol in blst:
        dlina += (ord(symbol)-1039)
print("Хэш исходного сообщения: ", dlina)
cipher = ""
# в этот файл записывается зашифрованный текст
f = open('shifr.txt', 'wt', encoding='utf-8')
final = ""
rassh = ""
decipher = 0
cipher1 = 0
rassh1 = 0
# шифрование и расшифрование
for symbol in text:
    if symbol == " ":
        final += " "
        rassh += " "
    elif symbol in blst:
        if ord(symbol) < 1056: # если буквы от А до П включительно
            ciph = (ord(symbol)%32) - 15 # с компьютерных символов букв переводим на те,
которыми вычисляем мы
            cipher = (ciph**E) % N # шифрование
            cipher1 += (ciph**E) % N # хэш зашифрования
        else:
            ciph = (ord(symbol)%32) + 17 # если буквы от Р до Я
            cipher = ciph**E % N # шифрование
            cipher1 += (ciph**E) % N # хэш зашифрования
        final += str(cipher) + "_" # формируем зашифрованное сообщение
        decipher = chr(((cipher **D) % N)%32 - 1 + ord('А')) # расшифрование
        rassh += str(decipher) # формируем расшифрованное сообщение
```

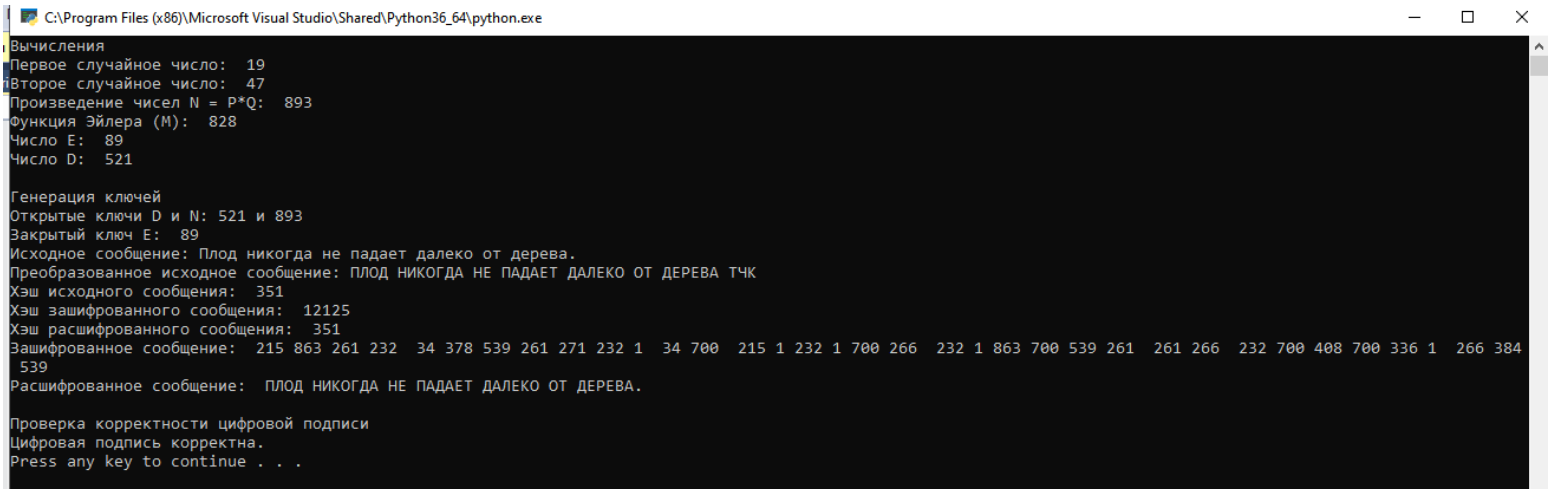
```
        rassh1 += (((cipher **D) % N)) # хэш расшифрования (сумма всех номеров букв
исходного сообщения)
final = final.replace('_', ' ')
print("Хэш зашифрованного сообщения: ", cipher1)
print("Хэш расшифрованного сообщения: ", rassh1)
print("Зашифрованное сообщение: ", final)
rassh = rassh.replace('Ц', 'Я') #
rassh = rassh.replace('_', '') # Если в сообщении попадется точка, она заменется на тчк
rassh = rassh.replace(' ТЧК', '.') # Если в сообщении попадется точка, она заменется на
тчк
rassh = rassh.replace(' ЗПТ', ',') # Если в сообщении попадется запятая, она заменется на
зпт
rassh = rassh.replace('ТИРЕ', '-') # Если в сообщении попадется - (тире), оно заменется
на тире
print("Расшифрованное сообщение: ", rassh,"\n")
f.writelines(final)
f.close()
# в этот файл записывается расшифрованный текст
file = open('rasshifr.txt', 'wt', encoding='utf-8')
file.writelines(rassh)
file.close()
# Проверка корректности цифровой подписи
print("Проверка корректности цифровой подписи")
if rassh1 == dlina:
    print("Цифровая подпись корректна.")
else:
    print("Цифровая подпись некорректна.")
    # https://stackoverflow.com/questions/21043075/generating-large-prime-numbers-in-
    python
```

5. Тестирование

Перед началом работы программы в файл “opentext.txt” записываем исходный текст.



Так выглядит окно выполнения программы. Ключ вводит пользователь



После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.

shifr.txt – Блокнот

Файл Правка Формат Вид Справка

```
215 863 261 232  34 378 539 261 271 232 1  34 700  215 1 232 1 700 266  232 1 863 700 539 261  261 266  232 700 408 700 336 1  266 384 539
```

rasshifr.txt – Блокнот

Файл Правка Формат Вид Справка

```
ПЛОД НИКОГДА НЕ ПАДАЕТ ДАЛЕКО ОТ ДЕРЕВА.
```

6. Работа с текстом не менее 1000 знаков

Перед началом работы программы в файл “opentext.txt” записываем исходный текст. (Полный исходный текст лежит в аннотации)

opentext.txt – Блокнот

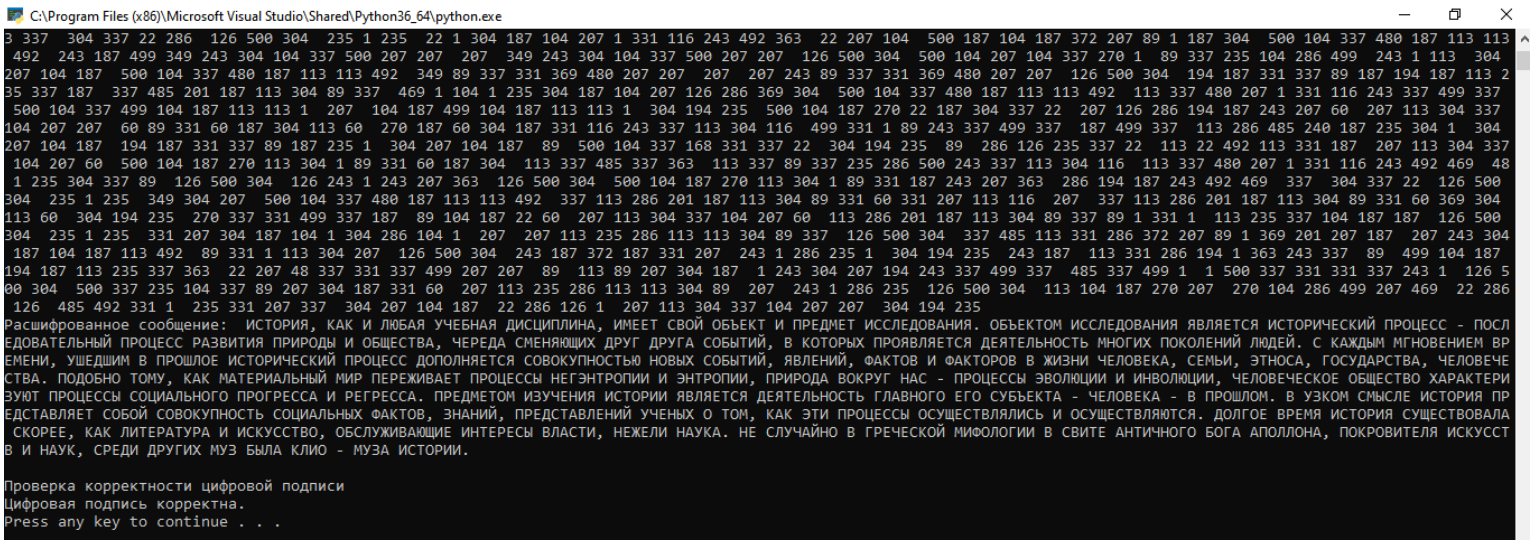
Файл Правка Формат Вид Справка

```
История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический
живающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств
```

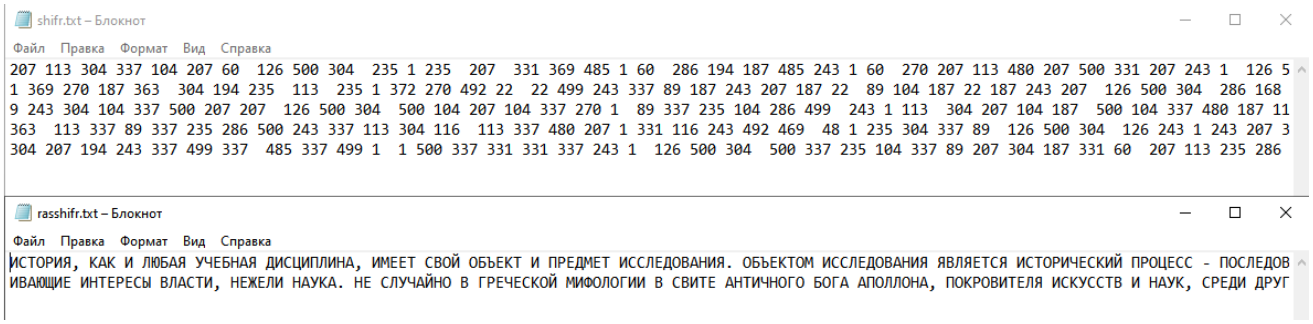
Так выглядит окно выполнения программы.

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe

```
Вычисления
Первое случайное число:  29
Второе случайное число:  19
Произведение чисел N = P*Q:  551
Функция Эйлера (M):  504
Число E:  17
Число D:  89

Генерация ключей
Открытые ключи D и N: 89 и 551
Закрытый ключ E:  17
Исходное сообщение: История, как и любая учебная дисциплина, имеет свой объект и предмет исследования. Объектом исследования является исторический процесс - последовате
льный процесс развития природы и общества, череда сменяющих друг друга событий, в которых проявляется деятельность многих поколений людей. С каждым мгновением времени,
ушедшим в прошлое исторический процесс дополняется совокупностью новых событий, явлений, фактов и факторов в жизни человека, семьи, этноса, государства, человечества. П
одобно тому, как материальный мир переживает процессы негэнтропии и энтропии, природа вокруг нас - процессы эволюции и инволюции, человеческое общество характеризуют пр
оцессы социального прогресса и регресса. Предметом изучения истории является деятельность главного его субъекта - человека - в прошлом. В узком смысле история представл
яет собой совокупность социальных фактов, знаний, представлений ученых о том, как эти процессы осуществлялись и осуществляются. Долгое время история существовала скорее
, как литература и искусство, обслуживающие интересы власти, нежели наука. Не случайно в греческой мифологии в свите античного бога Аполлона, покровителя искусств и нау
к, среди других муз была Клио - муза истории.
Преобразованное исходное сообщение: ИСТОРИЯ ЗПТ КАК И ЛЮБАЯ УЧЕБНАЯ ДИСЦИПЛИНА ЗПТ ИМЕЕТ СВОЙ ОБЪЕКТ И ПРЕДМЕТ ИССЛЕДОВАНИЯ ТЧК ОБЪЕКТОМ ИССЛЕДОВАНИЯ ЯВЛЯЕТСЯ ИСТОРИЧЕС
КИЙ ПРОЦЕСС ТИРЕ ПОСЛЕДОВАТЕЛЬНЫЙ ПРОЦЕСС РАЗВИТИЯ ПРИРОДЫ И ОБЩЕСТВА ЗПТ ЧЕРЕДА СМЕНЯЮЩИХ ДРУГ ДРУГА СОБЫТИЙ ЗПТ В КОТОРЫХ ПРОЯВЛЯЕТСЯ ДЕЯТЕЛЬНОСТЬ МНОГИХ ПОКОЛЕНИЙ ЛЮ
ДЕЙ ТЧК С КАЖДЫМ МГНОВЕНИЕМ ВРЕМЕНИ ЗПТ УШЕДШИМ В ПРОШЛОЕ ИСТОРИЧЕСКИЙ ПРОЦЕСС ДОПОЛНЯЕТСЯ СОВОКУПНОСТЬЮ НОВЫХ СОБЫТИЙ ЗПТ ЯВЛЕНИЙ ЗПТ ФАКТОВ И ФАКТОРОВ В ЖИЗНИ ЧЕЛОВЕК
А ЗПТ СЕМЬИ ЗПТ ЭТНОСА ЗПТ ГОСУДАРСТВА ЗПТ ЧЕЛОВЕЧЕСТВА ТЧК ПОДОБНО ТОМУ ЗПТ КАК МАТЕРИАЛЬНЫЙ МИР ПЕРЕЖИВАЕТ ПРОЦЕССЫ НЕГЭНТРОПИИ И ЭНТРОПИИ ЗПТ ПРИРОДА ВОКРУГ НАС ТИРЕ
 ПРОЦЕССЫ ЭВОЛЮЦИИ И ИНВОЛЮЦИИ ЗПТ ЧЕЛОВЕЧЕСКОЕ ОБЩЕСТВО ХАРАКТЕРИЗУЮТ ПРОЦЕССЫ СОЦИАЛЬНОГО ПРОГРЕССА И РЕГРЕССА ТЧК ПРЕДМЕТОМ ИЗУЧЕНИЯ ИСТОРИИ ЯВЛЯЕТСЯ ДЕЯТЕЛЬНОСТЬ ГЛ
АВНОГО ЕГО СУБЪЕКТА ТИРЕ ЧЕЛОВЕКА ТИРЕ В ПРОШЛОМ ТЧК В УЗКОМ СМЫСЛЕ ИСТОРИЯ ПРЕДСТАВЛЯЕТ СОБОЙ СОВОКУПНОСТЬ СОЦИАЛЬНЫХ ФАКТОВ ЗПТ ЗНАНИЙ ЗПТ ПРЕДСТАВЛЕНИЙ УЧЕНЫХ О ТОМ
ЗПТ КАК ЭТИ ПРОЦЕССЫ ОСУЩЕСТВЛЯЛИСЬ И ОСУЩЕСТВЛЯЮТСЯ ТЧК ДОЛГОЕ ВРЕМЯ ИСТОРИЯ СУЩЕСТВОВАЛА СКОРЕЕ ЗПТ КАК ЛИТЕРАТУРА И ИСКУССТВО ЗПТ ОБСЛУЖИВАЮЩИЕ ИНТЕРЕСЫ ВЛАСТИ ЗПТ Н
ЕЖЕЛИ НАУКА ТЧК НЕ СЛУЧАЙНО В ГРЕЧЕСКОЙ МИФОЛОГИИ В СВИТЕ АНТИЧНОГО БОГА АПОЛЛОНА ЗПТ ПОКРОВИТЕЛЯ ИСКУССТВ И НАУК ЗПТ СРЕДИ ДРУГИХ МУЗ БЫЛА КЛИО ТИРЕ МУЗА ИСТОРИИ ТЧК
Хэш исходного сообщения:  15239
Хэш зашифрованного сообщения:  260229
Хэш расшифрованного сообщения:  15239
Зашифрованное сообщение:  207 113 304 337 104 207 60  126 500 304  235 1 235  207  331 369 485 1 60  286 194 187 485 243 1 60  270 207 113 480 207 500 331 207 243 1  12
6 500 304  207 22 187 187 304  113 89 337 363  337 485 240 187 235 304  207  500 104 187 270 22 187 304  207 113 113 331 187 270 337 89 1 243 207 60  304 194 235  337 4
85 240 187 235 304 337 22  207 113 113 331 187 270 337 89 1 243 207 60  60 89 331 60 187 304 113 60  207 113 304 337 104 207 194 187 113 235 207 363  500 104 337 480 18
7 113 113  304 207 104 187  500 337 113 331 187 270 337 89 1 304 187 331 116 243 492 363  500 104 337 480 187 113 113  104 1 126 89 207 304 207 60  500 104 207 104 337
270 492  207  337 485 201 187 113 304 89 1  126 500 304  194 187 104 187 270 1  113 22 187 243 60 369 201 207 469  270 104 286 499  270 104 286 499 1  113 337 485 492 3
04 207 363  126 500 304  89  235 337 304 337 104 492 469  500 104 337 60 89 331 60 187 304 113 60  270 187 60 304 187 331 116 243 337 113 304 116  22 243 337 499 207 46
9  500 337 235 337 331 187 243 207 363  331 369 270 187 363  304 194 235  113  235 1 372 270 492 22  22 499 243 337 89 187 243 207 187 22  89 104 187 22 187 243 207  12
6 500 304  286 168 187 270 168 207 22  89  500 104 337 168 331 337 187  207 113 304 337 104 207 194 187 113 235 207 363  500 104 337 480 187 113 113  270 337 500 337 33
1 243 60 187 304 113 60  113 337 89 337 235 286 500 243 337 113 304 116 369  243 337 89 492 469  113 337 485 492 304 207 363  126 500 304  60 89 331 187 243 207 363  12
6 500 304  48 1 235 304 337 89  207  48 1 235 304 337 104 337 89  89  372 207 126 243 207  194 187 331 337 89 187 235 1  126 500 304  113 187 22 116 207  126 500 304  3
49 304 243 337 113 1  126 500 304  499 337 113 286 270 1 104 113 304 89 1  126 500 304  194 187 331 337 89 187 194 187 113 304 89 1  304 194 235  500 337 270 337 485 24
3 337  304 337 22 286  126 500 304  235 1 235  22 1 304 187 104 207 1 331 116 243 492 363  22 207 104  500 187 104 187 372 207 89 1 187 304  500 104 337 480 187 113 113
 492  243 187 499 349 243 304 104 337 500 207 207  207  349 243 304 104 337 500 207 207  126 500 304  500 104 207 104 337 270 1  89 337 235 104 286 499  243 1 113  304
207 104 187  500 104 337 480 187 113 113 492  349 89 337 331 369 480 207 207  207  207 243 89 337 331 369 480 207 207  126 500 304  194 187 331 337 89 187 194 187 113 2
```



После выполнения программы в файл “shifr.txt” записывается зашифрованный текст, а в файл “rasshifr.txt” расшифрованный текст.



Полный зашифрованный текст:

207 113 304 337 104 207 60 126 500 304 235 1 235 207 331 369 485 1 60 286 194 187 485 243 1 60 270 207 113 480 207 500 331 207 243 1 126 500 304 207 22 187 187 304 113 89 337 363 337 485 240 187 235 304 207 500 104 187 270 22 187 304 207 113 113 331 187 270 337 89 1 243 207 60 304 194 235 337 485 240 187 235 304 337 22 207 113 113 331 187 270 337 89 1 243 207 60 60 89 331 60 187 304 113 60 207 113 304 337 104 207 194 187 113 235 207 363 500 104 337 480 187 113 113 304 207 104 187 500 337 113 331 187 270 337 89 1 304 187 331 116 243 492 363 500 104 337 480 187 113 113 104 1 126 89 207 304 207 60 500 104 207 104 337 270 492 207 337 485 201 187 113 304 89 1 126 500 304 194 187 104 187 270 1 113 22 187 243 60 369 201 207 469 270 104 286 499 270 104 286 499 1 113 337 485 492 304 207 363 126 500 304 89 235 337 304 337 104 492 469 500 104 337 60 89 331 60 187 304 113 60 270 187 60 304 187 331 116 243 337 113 304 116 22 243 337 499 207 469 500 337 235 337 331 187 243 207 363 331 369 270 187 363 304 194 235 113 235 1 372 270 492 22 22 499 243 337 89 187 243 207 187 22 89 104 187 22 187 243 207 126 500 304 286 168 187 270 168 207 22 89 500 104 337 168 331 337 187 207 113 304 337 104 207 194 187 113 235 207 363 500 104 337 480 187 113 113 270 337 500 337 331 243 60 187 304 113 60 113 337 89 337 235 286 500 243 337 113 304 116 369 243 337 89 492 469 113 337 485 492 304 207 363 126 500 304

60 89 331 187 243 207 363 126 500 304 48 1 235 304 337 89 207 48 1 235 304 337 104 337 89 89 372 207 126 243 207 194 187 331 337 89 187 235 1 126 500 304 113 187 22 116 207 126 500 304 349 304 243 337 113 1 126 500 304 499 337 113 286 270 1 104 113 304 89 1 126 500 304 194 187 331 337 89 187 194 187 113 304 89 1 304 194 235 500 337 270 337 485 243 337 304 337 22 286 126 500 304 235 1 235 22 1 304 187 104 207 1 331 116 243 492 363 22 207 104 500 187 104 187 372 207 89 1 187 304 500 104 337 480 187 113 113 492 243 187 499 349 243 304 104 337 500 207 207 207 349 243 304 104 337 500 207 207 126 500 304 500 104 207 104 337 270 1 89 337 235 104 286 499 243 1 113 304 207 104 187 500 104 337 480 187 113 113 492 349 89 337 331 369 480 207 207 207 207 243 89 337 331 369 480 207 207 126 500 304 194 187 331 337 89 187 194 187 113 235 337 187 337 485 201 187 113 304 89 337 469 1 104 1 235 304 187 104 207 126 286 369 304 500 104 337 480 187 113 113 492 113 337 480 207 1 331 116 243 337 499 337 500 104 337 499 104 187 113 113 1 207 104 187 499 104 187 113 113 1 304 194 235 500 104 187 270 22 187 304 337 22 207 126 286 194 187 243 207 60 207 113 304 337 104 207 207 60 89 331 60 187 304 113 60 270 187 60 304 187 331 116 243 337 113 304 116 499 331 1 89 243 337 499 337 187 499 337 113 286 485 240 187 235 304 1 304 207 104 187 194 187 331 337 89 187 235 1 304 207 104 187 89 500 104 337 168 331 337 22 304 194 235 89 286 126 235 337 22 113 22 492 113 331 187 207 113 304 337 104 207 60 500 104 187 270 113 304 1 89 331 60 187 304 113 337 485 337 363 113 337 89 337 235 286 500 243 337 113 304 116 113 337 480 207 1 331 116 243 492 469 48 1 235 304 337 89 126 500 304 126 243 1 243 207 363 126 500 304 500 104 187 270 113 304 1 89 331 187 243 207 363 286 194 187 243 492 469 337 304 337 22 126 500 304 235 1 235 349 304 207 500 104 337 480 187 113 113 492 337 113 286 201 187 113 304 89 331 60 331 207 113 116 207 337 113 286 201 187 113 304 89 331 60 369 304 113 60 304 194 235 270 337 331 499 337 187 89 104 187 22 60 207 113 304 337 104 207 60 113 286 201 187 113 304 89 337 89 1 331 1 113 235 337 104 187 187 126 500 304 235 1 235 331 207 304 187 104 1 304 286 104 1 207 207 113 235 286 113 113 304 89 337 126 500 304 337 485 113 331 286 372 207 89 1 369 201 207 187 207 243 304 187 104 187 113 492 89 331 1 113 304 207 126 500 304 243 187 372 187 331 207 243 1 286 235 1 304 194 235 243 187 113 331 286 194 1 363 243 337 89 499 104 187 194 187 113 235 337 363 22 207 48 337 331 337 499 207 207 89 113 89 207 304 187 1 243 304 207 194 243 337 499 337 485 337 499 1 1 500 337 331 331 337 243 1 126 500 304 500 337 235 104 337 89 207 304 187 331 60 207 113 235 286 113 113 304 89 207 243 1 286 235 126 500 304 113 104 187 270 207 270 104 286 499 207 469 22 286 126 485 492 331 1 235 331 207 337 304 207 104 187 22 286 126 1 207 113 304 337 104 207 207 304 194 235

Полный расшифрованный текст:

ИСТОРИЯ, КАК И ЛЮБАЯ УЧЕБНАЯ ДИСЦИПЛИНА, ИМЕЕТ СВОЙ ОБЪЕКТ И ПРЕДМЕТ ИССЛЕДОВАНИЯ. ОБЪЕКТОМ ИССЛЕДОВАНИЯ

ЯВЛЯЕТСЯ ИСТОРИЧЕСКИЙ ПРОЦЕСС - ПОСЛЕДОВАТЕЛЬНЫЙ ПРОЦЕСС РАЗВИТИЯ ПРИРОДЫ И ОБЩЕСТВА, ЧЕРЕДА СМЕНЯЮЩИХ ДРУГ ДРУГА СОБЫТИЙ, В КОТОРЫХ ПРОЯВЛЯЕТСЯ ДЕЯТЕЛЬНОСТЬ МНОГИХ ПОКОЛЕНИЙ ЛЮДЕЙ. С КАЖДЫМ МГНОВЕНИЕМ ВРЕМЕНИ, УШЕДШИМ В ПРОШЛОЕ ИСТОРИЧЕСКИЙ ПРОЦЕСС ДОПОЛНЯЕТСЯ СОВОКУПНОСТЬЮ НОВЫХ СОБЫТИЙ, ЯВЛЕНИЙ, ФАКТОВ И ФАКТОРОВ В ЖИЗНИ ЧЕЛОВЕКА, СЕМЬИ, ЭТНОСА, ГОСУДАРСТВА, ЧЕЛОВЕЧЕСТВА. ПОДОБНО ТОМУ, КАК МАТЕРИАЛЬНЫЙ МИР ПЕРЕЖИВАЕТ ПРОЦЕССЫ НЕГЭНТРОПИИ И ЭНТРОПИИ, ПРИРОДА ВОКРУГ НАС - ПРОЦЕССЫ ЭВОЛЮЦИИ И ИНВОЛЮЦИИ, ЧЕЛОВЕЧЕСКОЕ ОБЩЕСТВО ХАРАКТЕРИЗУЮТ ПРОЦЕССЫ СОЦИАЛЬНОГО ПРОГРЕССА И РЕГРЕССА. ПРЕДМЕТОМ ИЗУЧЕНИЯ ИСТОРИИ ЯВЛЯЕТСЯ ДЕЯТЕЛЬНОСТЬ ГЛАВНОГО ЕГО СУБЪЕКТА - ЧЕЛОВЕКА - В ПРОШЛОМ. В УЗКОМ СМЫСЛЕ ИСТОРИЯ ПРЕДСТАВЛЯЕТ СОБОЙ СОВОКУПНОСТЬ СОЦИАЛЬНЫХ ФАКТОВ, ЗНАНИЙ, ПРЕДСТАВЛЕНИЙ УЧЕНЫХ О ТОМ, КАК ЭТИ ПРОЦЕССЫ ОСУЩЕСТВЛЯЛИСЬ И ОСУЩЕСТВЛЯЮТСЯ. ДОЛГОЕ ВРЕМЯ ИСТОРИЯ СУЩЕСТВОВАЛА СКОРЕЕ, КАК ЛИТЕРАТУРА И ИСКУССТВО, ОБСЛУЖИВАЮЩИЕ ИНТЕРЕСЫ ВЛАСТИ, НЕЖЕЛИ НАУКА. НЕ СЛУЧАЙНО В ГРЕЧЕСКОЙ МИФОЛОГИИ В СВИТЕ АНТИЧНОГО БОГА АПОЛЛОНА, ПОКРОВИТЕЛЯ ИСКУССТВ И НАУК, СРЕДИ ДРУГИХ МУЗ БЫЛА КЛИО - МУЗА ИСТОРИИ.

7. Вся работа происходит в файлах: “$_{mkiu}$opentext.txt”, “shifr.txt”, “rasshifr.txt”.

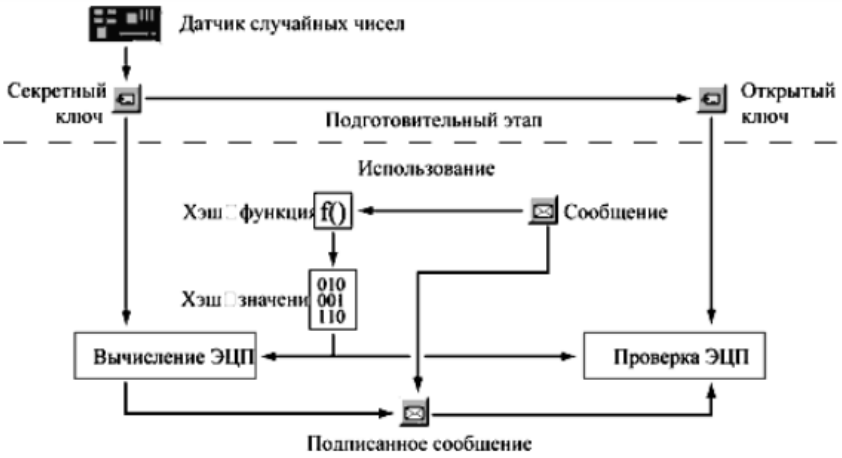
El Gamal.

1. Описание шифра.

Схема Эль-Гамаля (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94).

Цифровая подпись служит для того чтобы можно было установить изменения данных и чтобы установить подлинность подписавшейся стороны. Получатель подписанного сообщения может использовать цифровую подпись для доказательства третьей стороне того, что подпись действительно сделана отправляющей стороной. При работе в режиме подписи предполагается наличие фиксированной хеш-функции h(), значения которой лежат в интервале (1, p-1).

2. Алгоритм шифра.



Для того чтобы генерировать пару ключей (открытый ключ - секретный ключ), сначала выбирают некоторое **большое простое целое число P и большое целое число G, причем G < P**. Отправитель и получатель подписанного документа используют при вычислениях близкие большие целые числа **P** ($\sim 10^{308}$ или $\sim 2^{1024}$) и **G** ($\sim 10^{154}$ или $\sim 2^{512}$), которые **не являются секретными**.

1. Отправитель выбирает случайное целое число **X,**

$$1 < X \leq (P-1),$$

и вычисляет

$$Y = G^X \bmod P.$$

Y - открытый ключ.
X - секретный ключ.

2. Для того чтобы подписать сообщение M, сначала отправитель хэширует его с помощью хэш-функции h() в целое число m:

$$m = h(M),\ 1<m<(P-1),$$

и генерирует случайное целое число **K**, $1 < K < (P-1)$, такое, что **K и (P-1) являются взаимно простыми**.

Подписание:
отправитель вычисляет целое число **a**:

$$a = G^K \bmod P$$

и, применяя **расширенный алгоритм Евклида**, вычисляет с помощью секретного ключа X целое число **b** из уравнения

$$m = X * a + K * b \ (\text{mod } (P-1)).$$

Пара чисел (a,b) образует цифровую подпись S:

$$S = (a, b),$$

проставляемую под документом M.

3. (M, a, b) передается получателю.

Проверка: соответствует ли подпись S = (a, b) сообщению M.
получатель сначала вычисляет хэш по M:

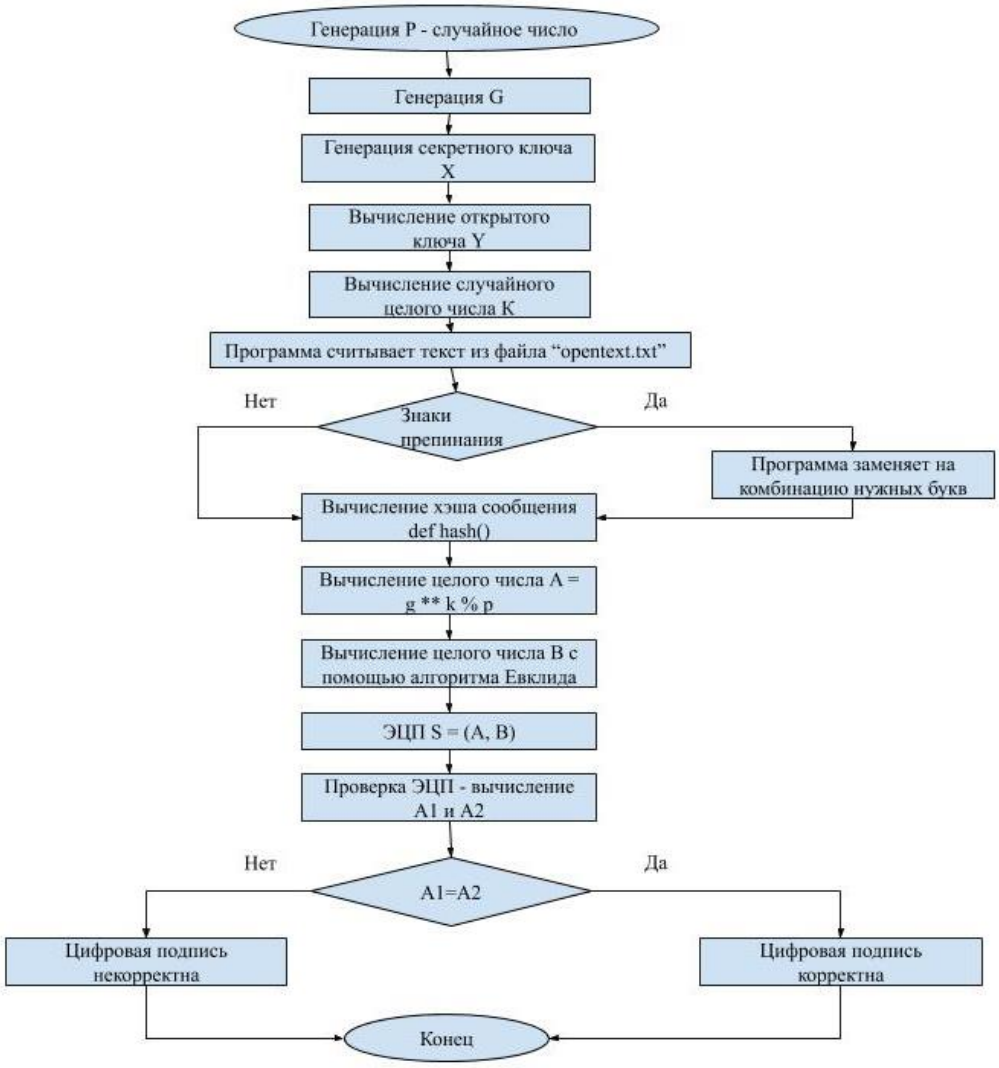
$$m = h(M),$$

Затем вычисляет значения

$$A1 = Y^a a^b \ (\text{mod } P) \text{ и } A2 = G^m \ (\text{mod } P).$$

Если **A1=A2** . т.е. $Y^a a^b \ (\text{mod } P) = G^m \ (\text{mod } P)$,
подпись верна – сообщение подлинное.

3. Блок-схема программы



4. Код программы

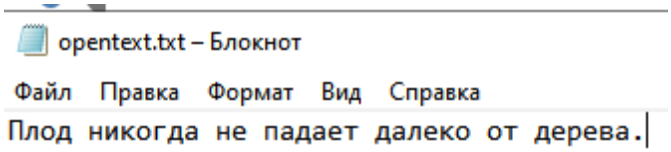
```
from random import *

# генерация P - открытый ключ - простое большое число
def rand_P():
    while True:
        p = randint(9999, 99999)
        if p % 2 != 0 and all(p % n != 0 for n in range(3, int((p ** 0.5) + 1), 2)):
            return p

# большое целое число G, причем G < P - открытый ключ
def rand_G(p):
    while True:
        g = randint(9999, 99999)
        if (g % 2 != 0) and all(g % n != 0 for n in range(3, int((g ** 0.5) + 1))) and g < p:
            return g + 1

# случайное целое число X: 1 < X ≤ (P-1) - секретный ключ
def rand_X(p):
    while True:
        x = randint(9999, 99999)
        if (x % 2 != 0) and all(x % n != 0 for n in range(3, int((x ** 0.5) + 1))) and x < p:
            return x + 1

# хэш функция - произвольная
def hash(text):
    h = 0
    for i in text:
        h += ord(i) % 500
    return h
```

```
# случайное целое число K, 1 < K < (P-1), такое, что K и (P-1) являются взаимно простыми
def NOD(a, b):
    while a != 0 and b != 0:
        if a > b:
            a %= b
        else:
            b %= a
    return a+b

def k_number(p):
    k = randrange(999, 9997, 2)
    while not NOD(k,p-1) == 1:
        k = randrange(999, 9997, 2)
    return k

f = open(r"opentext.txt", "rt", encoding='utf-8')
text = f.read()
print("Исходное сообщение:", text)
text = text.upper()
text = text.replace('.', ' ТЧК') # Если в сообщении попадется точка, она заменется на тчк
text = text.replace(',', ' ЗПТ') # Если в сообщении попадется запятая, она заменется на зпт
text = text.replace('-', 'ТИРЕ') # Если в сообщении попадется - (тире), оно заменется на тире

p = rand_P()
print("Первое случайное число P: ", p)

g = rand_G(p)
print("Число G: ", g)

x = rand_X(p)
print('Закрытый ключ X: ', x)
```

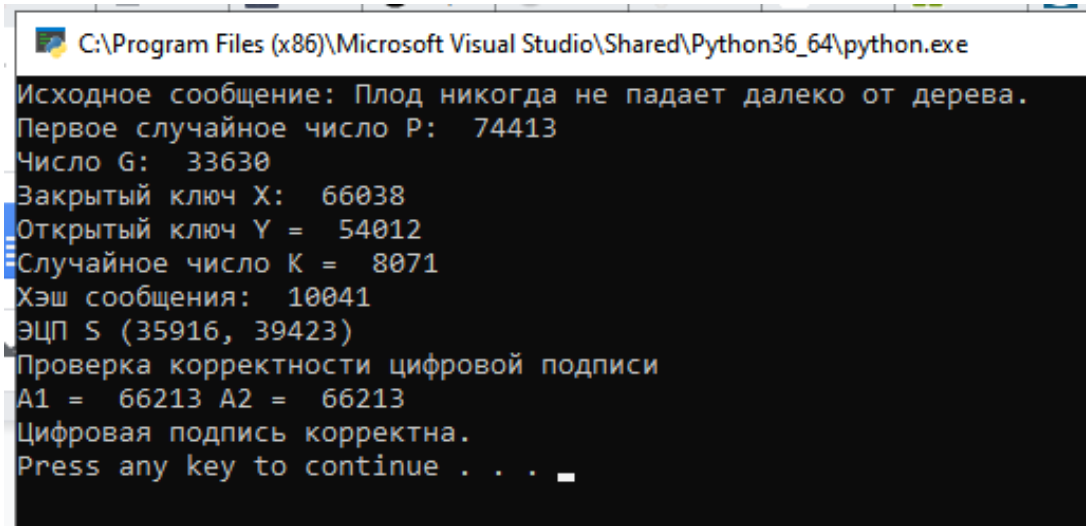
```
# Y - открытый ключ
y = g ** x % p
print('Открытый ключ Y = ', y)

k = k_number(p)
print('Случайное число K = ', k)

m = hash(text)
print("Хэш сообщения: ", m)
# вычисление целого числа A
a = g ** k % p

# вычисление B
b = 0
while m != (x * a + k * b) % (p-1):
    b += 1

print('ЭЦП S', (a,b))
# Проверка вычисление
a1 = (y ** a) * (a ** b) % p

a2 = g ** m % p
# Проверка корректности цифровой подписи
print("Проверка корректности цифровой подписи")
print('A1 = ', a1, 'A2 = ', a2)
if a1 == a2:
    print("Цифровая подпись корректна.")
else:
    print("Цифровая подпись некорректна.")
```

5. Тестирование

Перед началом работы программы в файл “opentext.txt” записываем исходный текст.



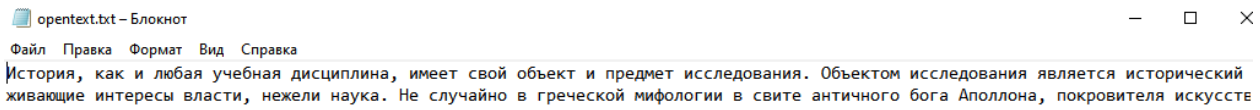
Так выглядит окно выполнения программы. Все числа генерируются программно.



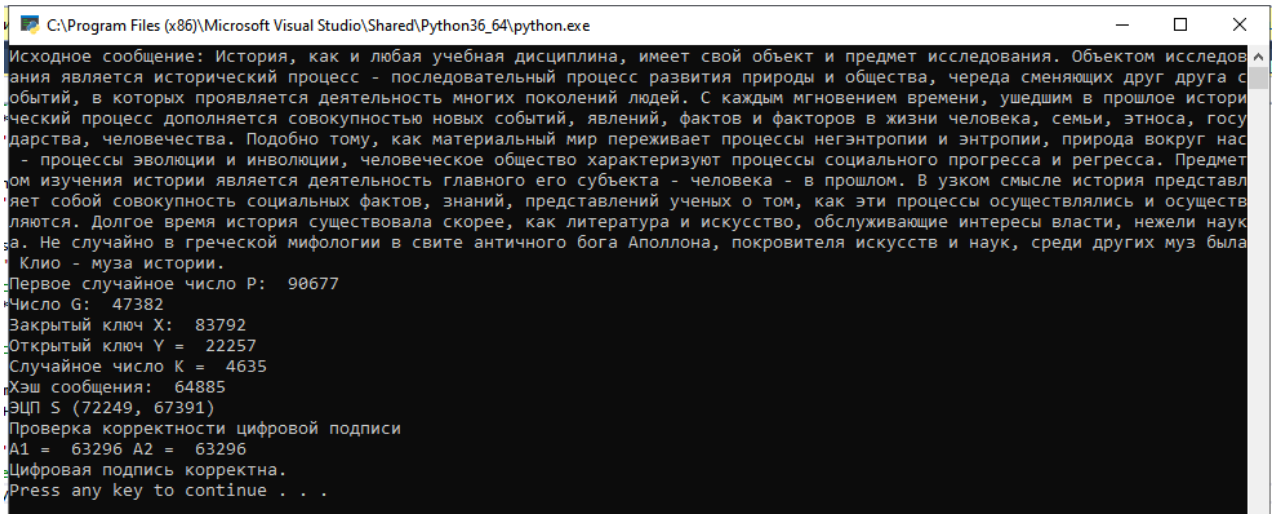
После выполнения программы в консоле высвечиваются значения открытых/закрытых ключей, электронно-цифровая подпись и сведения о проверке.

6. Работа с текстом не менее 1000 знаков

Перед началом работы программы в файл “opentext.txt” записываем исходный текст. (Полный исходный текст лежит в аннотации)



Так выглядит окно выполнения программы.



После выполнения программы в консоле высвечиваются значения открытых/закрытых ключей, электронно-цифровая подпись и сведения о проверке.

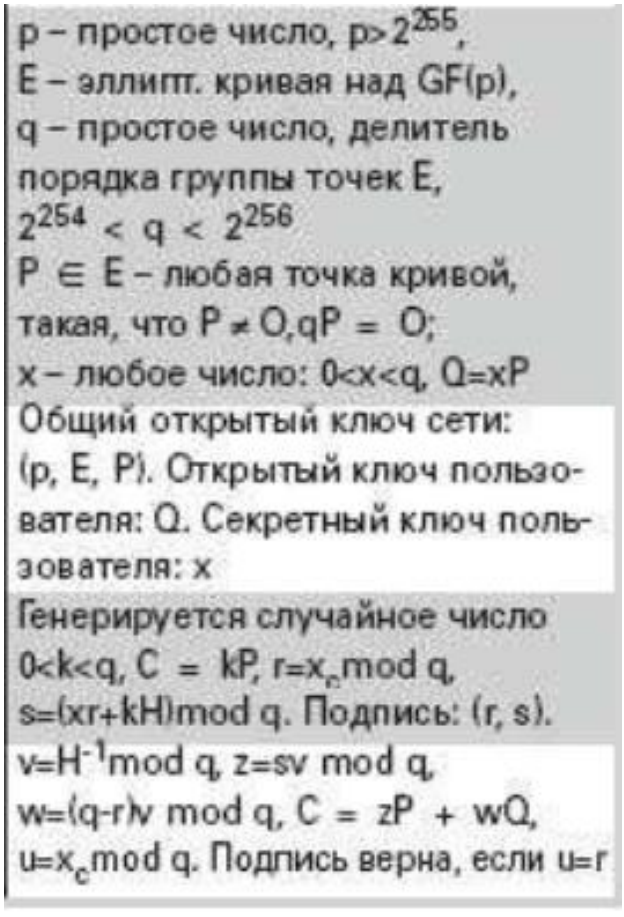
7. Вся работа происходит в файле “opentext.txt”.

ГОСТ Р 34.10-2012.

1. Описание шифра.

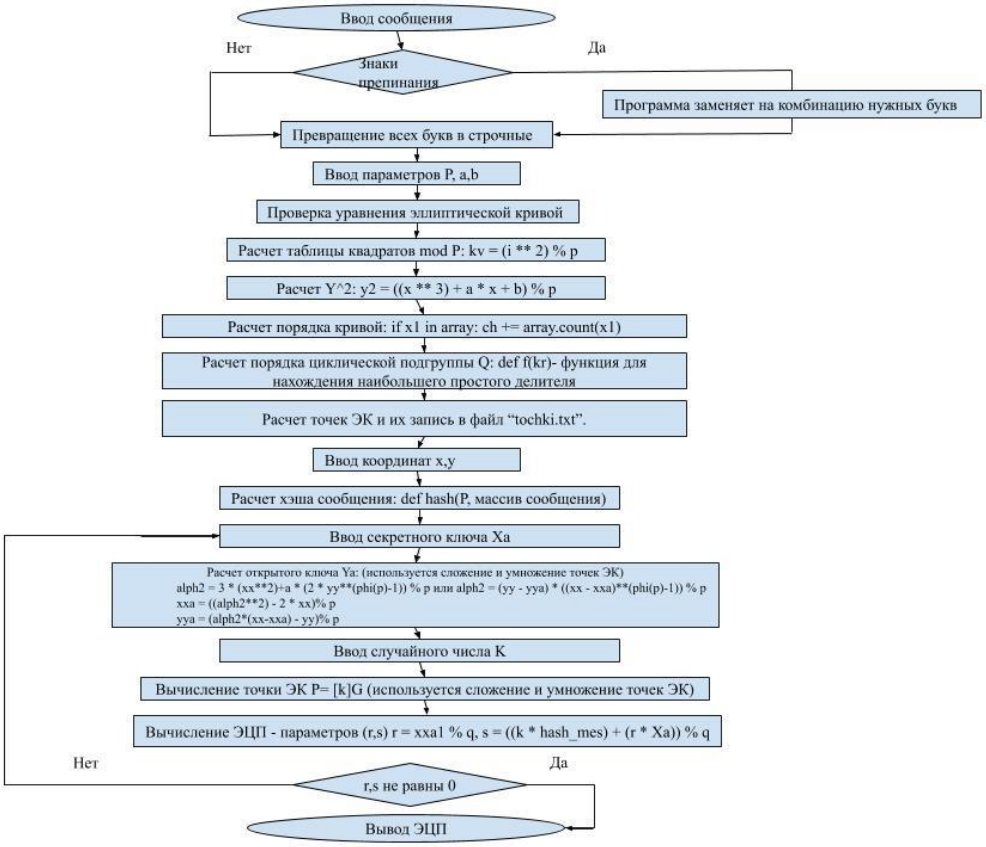
ГОСТ Р 34.10-2012 (полное название: «ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи») — российский стандарт, описывающий алгоритмы формирования и проверки электронной цифровой подписи. Принят и введён в действие Приказом Федерального агентства по техническому регулированию и метрологии от 7 августа 2012 года № 215-ст вместо ГОСТ Р 34.10-2001. До ГОСТ Р 34.10-2001 действовал стандарт ГОСТ Р 34.10-94. Данный алгоритм разработан главным управлением безопасности связи Федерального агентства правительственной связи и информации при Президенте Российской Федерации при участии Всероссийского научно-исследовательского института стандартизации. Разрабатывался взамен ГОСТ Р 34.10-94 для обеспечения большей стойкости алгоритма.

2. Алгоритм шифра.

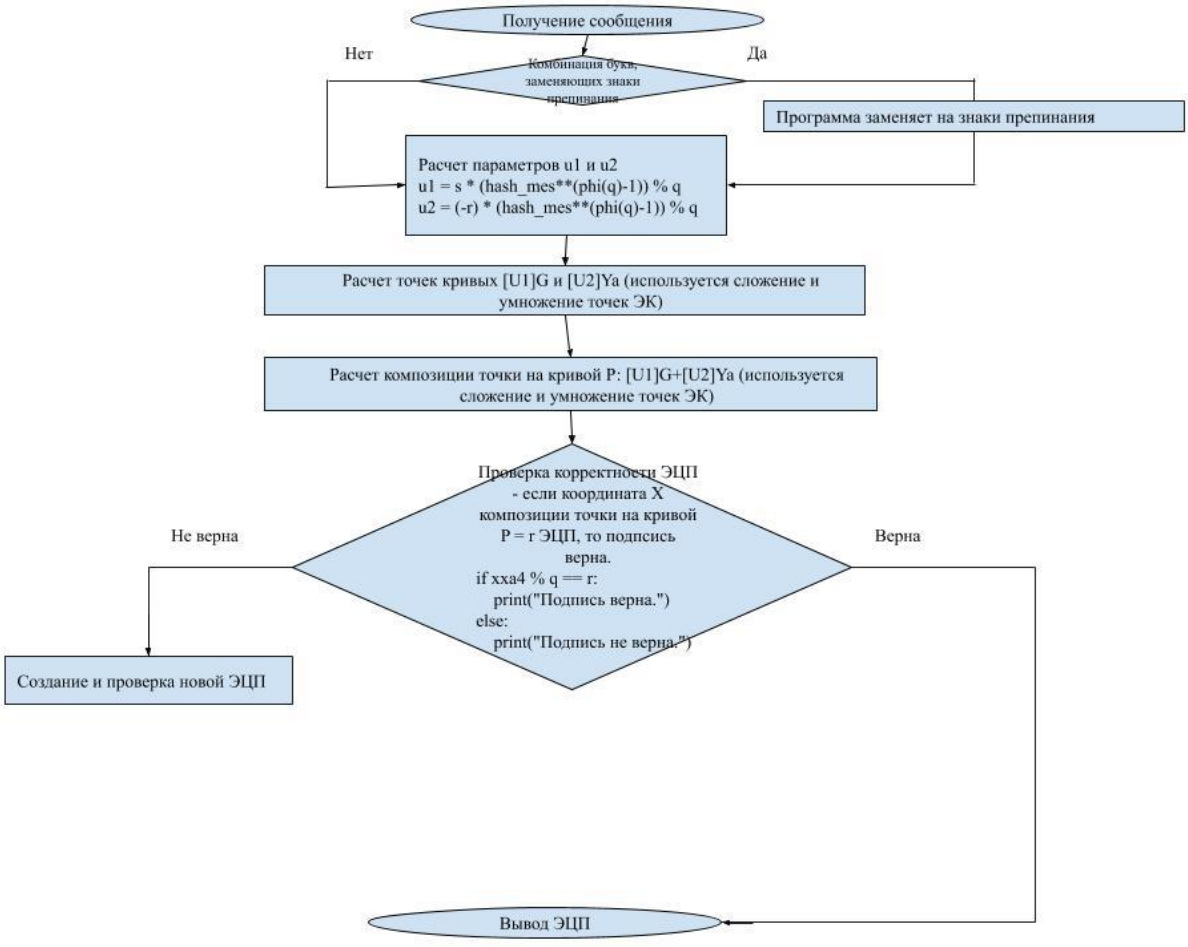


3. Блок-схема программы

Шифрование:



Проверка ЭЦП:



4. Код программы

```
from math import gcd
mes = input("Введите сообщение: ")
mes = mes.replace('.', ' тчк') # Если в сообщении попадется точка, она заменется на
тчк
mes = mes.replace(',', ' зпт') # Если в сообщении попадется запятая, она заменется на
зпт
mes = mes.replace('-', 'тире') # Если в сообщении попадется - (тире), оно заменется на
тире
mes = mes.replace(' ', 'прбл') #
mes = mes.lower()
print("Пользователь A, задайте параметры эллиптической кривой.")
p = input("Введите модуль P (P должно быть > 3 и простым числом): ")
p = int(p)
a = input("Введите паратметр A: ")
a = int(a)
b = input("Введите паратметр B: ")
b = int(b)
print("Уравнение эллиптической кривой: ", "y^2 = x^3 +",a,"* x ""+",b)
d = 4*(a**3)+27*(b**2) % p
if d == 0:
    print("Данные не верны, перезапустите программу и введите корректные данные.")
i = 0
print("Таблица квадратов P: ",)
array = []
array2 = []
for i in range(p):
    array2.append(i)
    kv = (i ** 2) % p
    i +=1
    array.append(kv)
print(array)
print("Y^2 = : ",)
array1 = []
for x in range(p):
    y2 = ((x ** 3) + a * x + b) % p
    x +=1
    array1.append(y2)
print(array1)
ch = 0
for x1 in array1:
    if x1 in array:
        ch += array.count(x1)
kr = ch+1
print("Порядок кривой: #E",p,"(",a,",",b,") =",kr)
# Порядок циклической подгруппы - функция для нахождения наибольшего простого делителя
def f(kr):
    pdelim = 2
    while kr > 1:
        if kr % pdelim == 0:
            kr = kr/pdelim
        else:
            pdelim+=1
    return pdelim
q = f(kr)
print("Порядок циклической подгруппы Q =", f(kr))
i3 = 0
fk = open('tochki.txt', 'w', encoding='utf-8') # открываем документ key.txt
fk = open('tochki.txt', 'a', encoding='utf-8') #
for s in range(p):
    for x1 in array1:
        if x1 in array:
            if (s ** 2) % p == x1:
                fk.write("(")
```

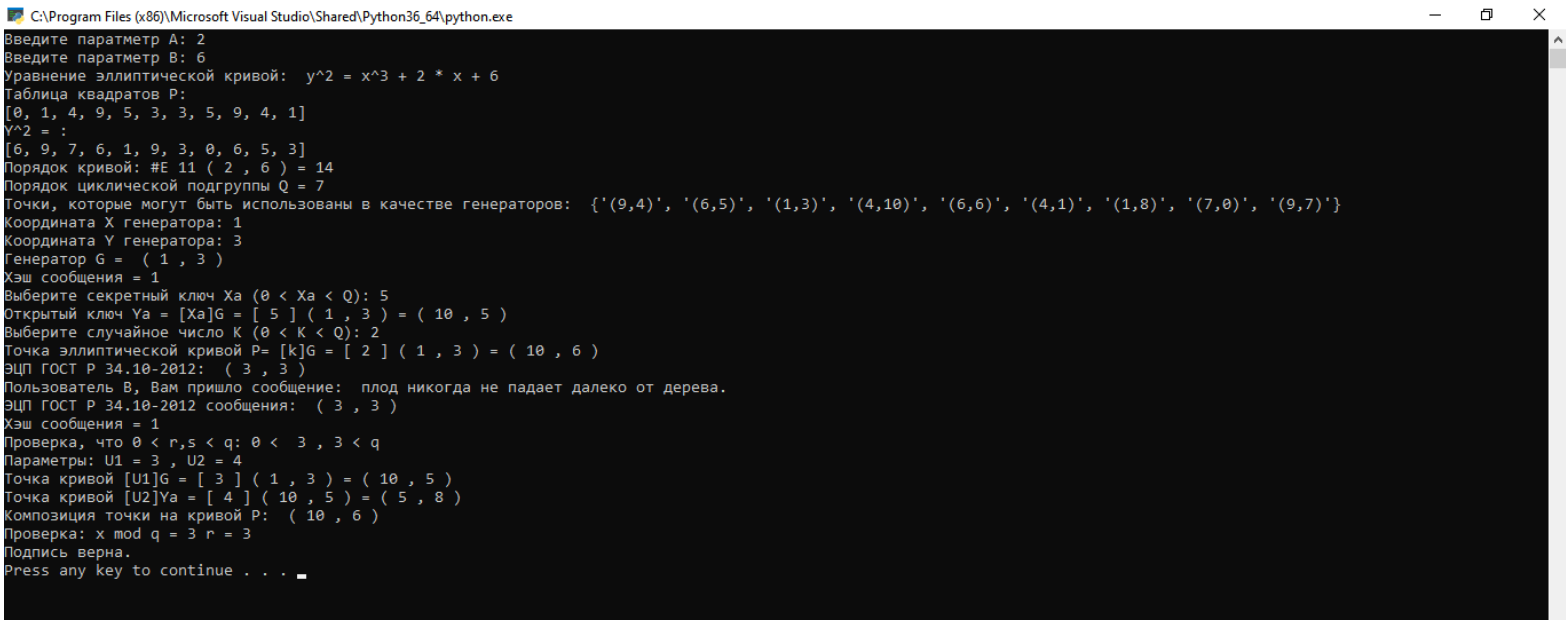
```
                fk.write(str(array1.index(x1)))
                fk.write(",")
                fk.write(str(s))
                fk.write(")")
                fk.write(" ")
fk.close()
fk = open('tochki.txt', 'r', encoding='utf-8') # открываем документ key.txt
tochki = fk.read() # считываем оттуда key - весь ключ
tochki = tochki.split()
tochki = set(tochki)
print("Точки, которые могут быть использованы в качестве генераторов: ",tochki)
xx = input("Координата X генератора: ")
xx = int(xx)
yy = input("Координата Y генератора: ")
yy = int(yy)
print("Генератор G = ","(",xx,",",yy,")")
# Хэш сообщения
h0 = 0
llst = {'а':1, 'б':2, 'в':3, 'г':4, 'д':5,
        'е':6, 'ж':7, 'з':8, 'и':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
        }
mes_list = list(mes)
llst_list = list()
for ii in range(len(mes_list)):
    llst_list.append(int(llst.get(mes_list[ii])))
print("Длина исходного сообщения {} символов".format(len(llst_list)))
def hash(mod,littlelist):
    i = 0
    h0 = 0
    while i < len(llst_list):
        if i == 0:
            h1 = ((h0 + int(llst_list[i]))**2) % p
        else:
            h1 = (((h1) + int(llst_list[i]))**2) % p
        i += 1
    return h1
hash_mes = hash(p, llst_list)
if hash_mes == 0:
    hash_mes = 1
print("Хэш сообщения = {}".format(hash_mes))
# Секретный ключ Xa
Xa = input("Выберите секретный ключ Xa (0 < Xa < Q): ")
Xa = int(Xa)
while Xa > q:
    Xa = input("Введенное число некорректно. Выберите секретный ключ Xa (0 < Xa < Q):
")
    Xa = int(Xa)
# Открытый ключ Ya
pp = 0
  # Функция Эйлера
def phi(n):
    amount = 0
    for k in range(1, n + 1):
        if gcd(n, k) == 1:
            amount += 1
    return amount
while pp < (Xa-1):
    if pp == 0:
        alph = 3 * (xx**2)+a
        alph1 = 2 * yy
```

```
        alph2 = alph * (alph1**(phi(p)-1)) % p
        pp += 1
        xxa = ((alph2**2) - 2 * xx)% p
        yya = (alph2*(xx-xxa) - yy)% p
    else:
        alph = yy - yya
        alph1 = xx - xxa
        alph2 = alph * (alph1**(phi(p)-1)) % p
        pp += 1
        xxa = ((alph2**2) - xx - xxa)% p
        yya = (alph2*(xx-xxa) - yy)% p
print("Открытый ключ Ya = [Xa]G = ",Xa,"]", "(",xx,",",yy,") =","(",xxa,",",yya,")")
# Случайное число K
k = input("Выберите случайное число K (0 < K < Q): ")
k = int(k)
while k > q:
    k = input("Введенное число некорректно. Выберите случайное число K (0 < K < Q): ")
    k = int(k)
# Точка эллиптической кривой P
ppp = 0
while ppp < (k-1):
    if ppp == 0:
        alph = 3 * (xx**2)+a
        alph1 = 2 * yy
        alph2 = alph * (alph1**(phi(p)-1)) % p
        ppp += 1
        xxa1 = ((alph2**2) - 2 * xx)% p
        yya1 = (alph2*(xx-xxa1) - yy)% p
    else:
        alph = yy - yya1
        alph1 = xx - xxa1
        alph2 = alph * (alph1**(phi(p)-1)) % p
        ppp += 1
        xxa1 = ((alph2**2) - xx - xxa1)% p
        yya1 = (alph2*(xx-xxa1) - yy)% p
print("Точка эллиптической кривой P= [k]G = ",k,"]", "(",xx,",",yy,")
=","(",xxa1,",",yya1,")")
# R - 1 цифра ЭЦП
r = xxa1 % q
if r == 0:
    print("Перезапустите программу и выберите другое случайное число K.")
# S - 2 цифра ЭЦП
s = ((k * hash_mes) + (r * Xa)) % q
if s == 0:
    print("Перезапустите программу и выберите другое случайное число K.")
print("ЭЦП ГОСТ Р 34.10-2012: ", "(",r,",",s,")")
mes1 = mes
mes1 = mes1.replace(' зпт', ',') # Если в сообщении попадется запятая, она заменется
на зпт
mes1 = mes1.replace('тире', '-') # Если в сообщении попадется - (тире), оно заменется
на тире
mes1 = mes1.replace('прбл', ' ') #
mes1 = mes1.replace(' тчк', '.') # Если в сообщении попадется точка, она заменется на
тчк
print("Пользователь B, Вам пришло сообщение: ", mes1)
print("ЭЦП ГОСТ Р 34.10-2012 сообщения: ", "(",r,",",s,")")
print("Хэш сообщения = {}".format(hash_mes))
print("Проверка, что 0 < r,s < q: 0 < ", r,",", s, "< q")
# U1 and U2
u1 = s * (hash_mes**(phi(q)-1)) % q
u2 = (-r) * (hash_mes**(phi(q)-1)) % q
print("Параметры: U1 =", u1,",", "U2 =", u2)
# Композиция точки на кривой P
pppp = 0
# [U1]G
```

```
while pppp <= (u1-1):
    if pppp == 0:
        alph = 3 * (xx**2)+a
        alph1 = 2 * yy
        alph2 = alph * (alph1**(phi(p)-1)) % p
        pppp += 1
        xxa3 = ((alph2**2) - 2 * xx)% p
        yya3 = (alph2*(xx-xxa3) - yy)% p
    else:
        alph = yy - yya1
        alph1 = xx - xxa1
        alph2 = alph * (alph1**(phi(p)-1)) % p
        pppp += 1
        xxa3 = ((alph2**2) - xx - xxa3)% p
        yya3 = (alph2*(xx-xxa3) - yy)% p
print("Точка кривой [U1]G = [",u1,"]", "(",xx,",",yy,") =","(",xxa3,",",yya3,")")
# U2
ppppp = 0
while ppppp <= (u2-1):
    if ppppp == 0:
        alph = 3 * (xxa**2)+a
        alph1 = 2 * yya
        alph2 = alph * (alph1**(phi(p)-1)) % p
        ppppp += 1
        xxa2 = ((alph2**2) - 2 * xxa)% p
        yya2 = (alph2*(xxa-xxa2) - yya)% p
    else:
        alph = yya - yya2
        alph1 = xxa - xxa2
        alph2 = alph * (alph1**(phi(p)-1)) % p
        ppppp += 1
        xxa2 = ((alph2**2) - xxa - xxa2)% p
        yya2 = (alph2*(xxa-xxa2) - yya)% p
print("Точка кривой [U2]Ya = [",u2,"]", "(",xxa,",",yya,") =","(",xxa2,",",yya2,")")
if xxa3 == xxa2 and yya3 == yya2:
    alph = 3 * (xxa2**2)+a
    alph1 = 2 * yya2
    alph2 = alph * (alph1**(phi(p)-1)) % p
    xxa4 = ((alph2**2) - 2 * xxa2)% p
    yya4 = (alph2*(xxa2-xxa4) - yya2)% p
else:
    alph = yya3 - yya2
    alph1 = xxa3 - xxa2
    alph2 = alph * (alph1**(phi(p)-1)) % p
    xxa4 = ((alph2**2) - xxa3 - xxa2)% p
    yya4 = (alph2*(xxa2-xxa4) - yya2)% p
print("Композиция точки на кривой P: ","(",xxa4,",",yya4,")")
xp = xxa4 % q
print("Проверка: x mod q =",xp,"r =",r)
if xxa4 % q == r:
    print("Подпись верна.")
else:
    print("Подпись не верна.")
```

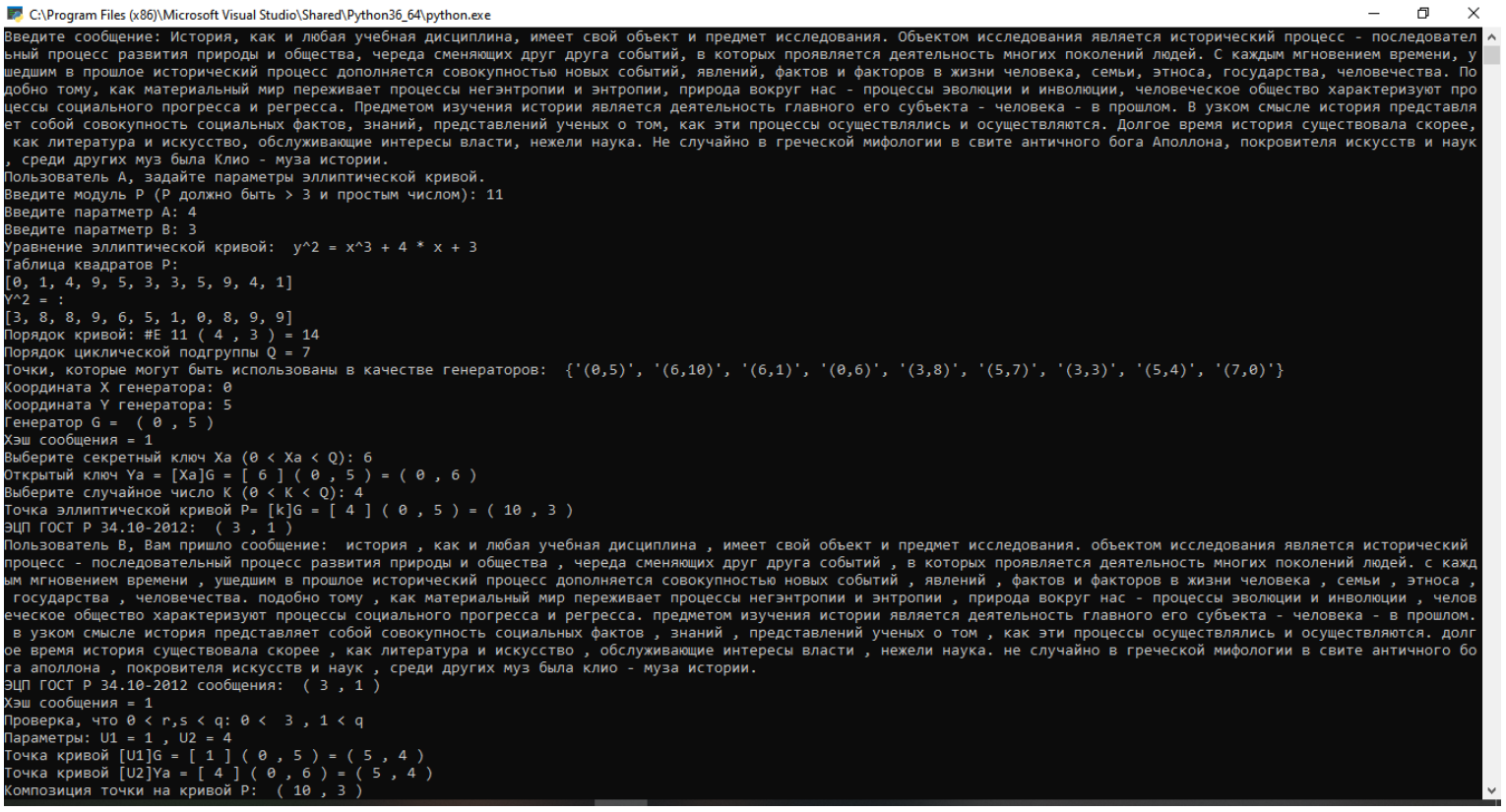
5. Тестирование

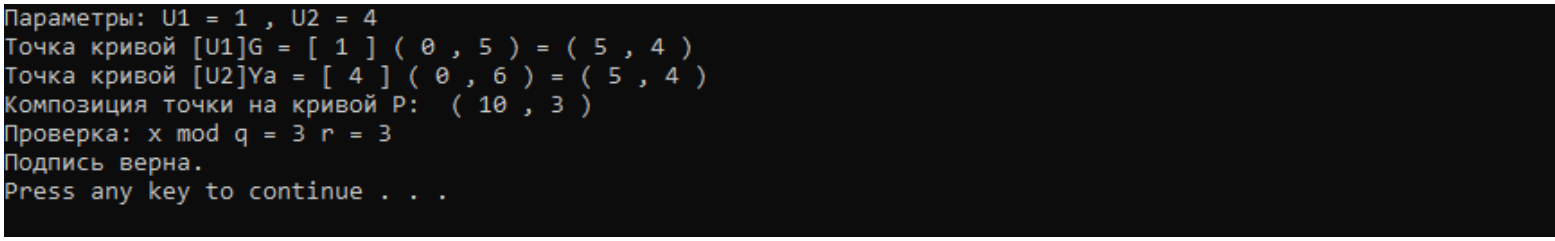
Так выглядит окно выполнения программы. Пользователь вводит открытый текст, а потом необходимые параметры: P, a, b, координаты точки (x, y).



6. Работа с текстом не менее 1000 знаков

Так выглядит окно выполнения программы.





7. Исполняемый файл

Вся работа происходит в окне выполнения программы.