Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Лабораторная работа №6. Компьютерная реализация электронной цифровой подписи

Выполнил: cтудент гр. 853501

Яковлев А.Б.

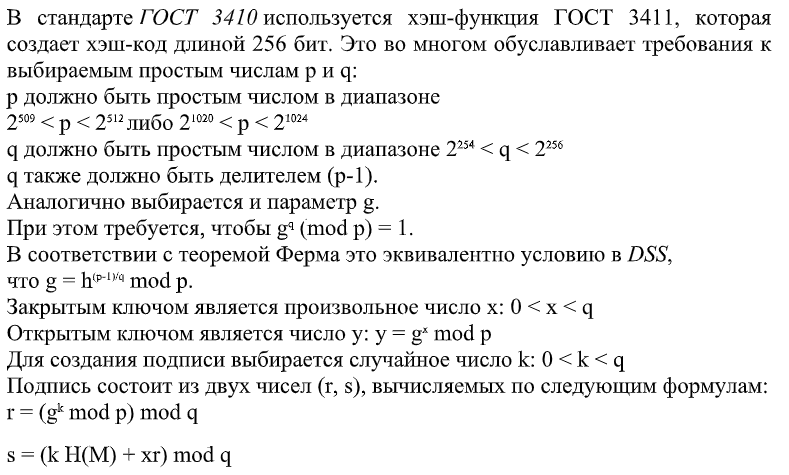
Проверил:

Протько М.И.

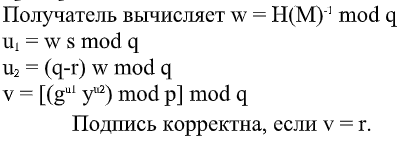
Минск 2021

# Постановка задачи и описание алгоритма

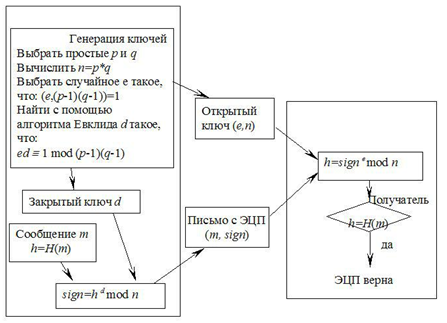
**Цель:** Реализовать программное средство формирования и проверки ЭЦП на базе алгоритма ГОСТ 3410.



Проверка подписи

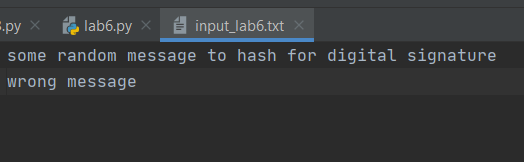


# Блок-схемы алгоритмов

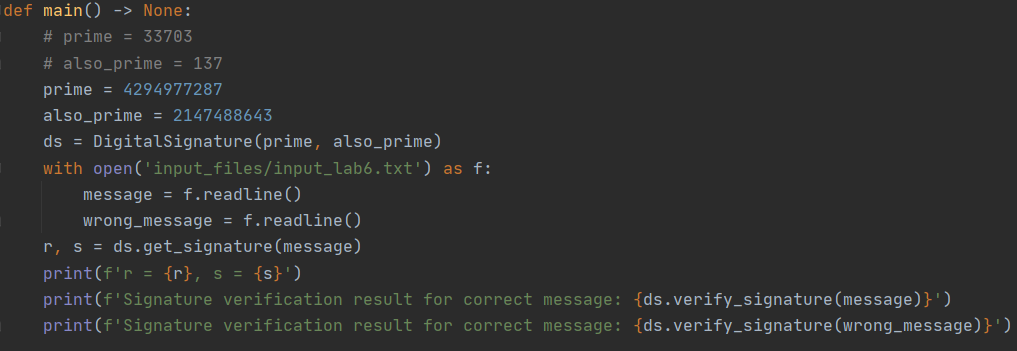


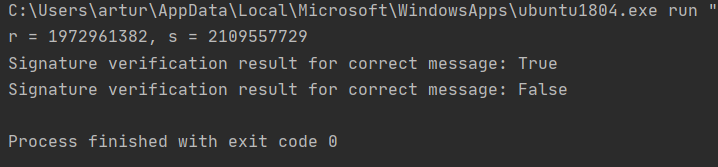
# Результаты выполнения программы

Входной файл



Результат выполнения





# Исходный код

import typing as tp  
from numpy.random import randint  
from functions import is\_prime, modular\_multiplicative\_inverse, get\_message\_hash, factorize  
  
  
class DigitalSignature:  
 def \_\_init\_\_(self, p: int, q: int) -> None:  
 *"""  
 These are preferable sizes, though not required* ***:param*** *p: prime number* ***:param*** *q: prime number, divider of p - 1  
 """* assert is\_prime(p)  
 assert is\_prime(q)  
 if (p - 1) % q != 0:  
 raise ValueError('q should be a divider of p - 1')  
 self.p = p  
 self.q = q  
 self.g = pow(modular\_multiplicative\_inverse(q, p), (p - 1) // q, p)  
 self.r = 0  
 self.s = 0  
 self.\_private\_key = randint(1, q)  
 self.\_session\_key = randint(1, q)  
 self.public\_key = pow(self.g, self.\_private\_key, self.p)  
  
 def get\_signature(self, message: str) -> tp.Tuple[int, int]:  
 hashed\_message = get\_message\_hash(message) % self.p  
 self.r = pow(self.g, self.\_session\_key, self.p) % self.q  
 self.s = (self.\_session\_key \* hashed\_message + self.\_private\_key \* self.r) % self.q  
 return self.r, self.s  
  
 def verify\_signature(self, message: str) -> bool:  
 hashed\_message = get\_message\_hash(message) % self.p  
 w = pow(hashed\_message, -1, self.q)  
 u1 = w \* self.s % self.q  
 u2 = (self.q - self.r) \* w % self.q  
 v = pow(self.g, u1, self.p) \* pow(self.public\_key, u2, self.p) % self.p % self.q  
 return v == self.r  
  
  
def main() -> None:  
 # prime = 33703  
 # also\_prime = 137  
 prime = 4294977287  
 also\_prime = 2147488643  
 ds = DigitalSignature(prime, also\_prime)  
 with open('input\_lab6.txt') as f:  
 message = f.readline()  
 wrong\_message = f.readline()  
 r, s = ds.get\_signature(message)  
 print(f'r = {r}, s = {s}')  
 print(f'Signature verification result for correct message: {ds.verify\_signature(message)}')  
 print(f'Signature verification result for correct message: {ds.verify\_signature(wrong\_message)}')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# Вывод

В результате выполнения работы была реализована программа, позволяющая формировать и проверять электронную цифровую подпись на основе алгоритма ГОСТ-3410.