Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Московский политехнический университет»

(Московский политех)

Отчёт по курсу «Программирование криптографических алгоритмов»

Лабораторная работа 9. Генерация цифровой подписи



Выполнил:

Студент группы 221-352

Иванов В. В.

Проверил преподаватель: Бутакова Н. Г.

Москва 2024г.

**Аннотация**

* **Среда программирования**
  + Visual Studio Code
* **Язык программирования**
  + Python
* **Процедуры для запуска программы** 
  + Visual Studio Code (main.py)
* **Пословица-тест**
  + Тот, кто ложится на два стула, падает на ребра.
* **Текст для проверки работы (не меньше 1000 знаков (1430))**

Жизнь - это удивительное приключение, полное разнообразных событий и встреч. В каждом моменте мы находим что-то новое и уникальное. Стремление к росту и саморазвитию вдохновляет нас на поиск новых горизонтов. Важно помнить, что каждый шаг вперед приносит с собой уроки и опыт.

Разнообразие культур, языков и традиций делает наш мир удивительно богатым. Общение с людьми разных национальностей расширяет кругозор, позволяя нам понимать и уважать друг друга. Взаимное уважение и терпимость создают основу для гармоничного сосуществования.

Природа тоже играет важную роль в нашей жизни. Красота закатов, шум океана, пение птиц - все это напоминает нам о величии мира природы. Забота о окружающей среде становится неотъемлемой частью ответственного образа жизни.

Работа и творчество придают смысл нашим усилиям. Стремление к достижению целей мотивирует нас на новые начинания. Каждый проект, даже самый маленький, приносит удовлетворение и чувство выполненного долга.

Семья и друзья являются надежной опорой в нашей жизни. Обмен историями, веселые посиделки и поддержка в трудные моменты создают теплую атмосферу взаимопонимания и любви.

Таким образом, наша жизнь - это мозаика различных моментов, соединенных воедино. Важно ценить каждый момент и стремиться делать мир вокруг нас ярче и лучше. С любовью, терпением и целеустремленностью мы можем создавать свою уникальную историю, наполненную смыслом и радостью.

**24.RSA DS**



**Блок-схема программы**

**Код программы с комментариями**

import math

from functions import alph, encodingFormat, isPrime, coprime, inputText, saveOutput

def hash(message, mod, alph):

    h = 0

    for letter in message:

        h = ((h + alph.index(letter) + 1) \*\* 2) % mod

    return h

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

def fi(n):

    num = 0

    for i in range(1, n):

        if gcd(i, n) == 1:

            num += 1

    return num

def comparison(comp):

    for y in range(comp[2]):

        if (comp[0] \* y) % comp[2] == comp[1] % comp[2]:

            return y

    return 0

def set\_autocomplete\_e(es):

    rsa\_e\_autocomplete = []

    for e in es:

        rsa\_e\_autocomplete.append(e)

    return rsa\_e\_autocomplete

def set\_d(fin, e):

    d = comparison([e, 1, fin])

    if d == e:

        return "D equals E, encryption is useless"

    return d

def RSA\_DS\_check\_parameters(p, q, e, ds):

    if math.isnan(p) or math.isnan(q):

        return "p or q is NaN"

    if not (isPrime(p) and isPrime(q)):

        return "p or q is not prime"

    if p == q:

        return "p == q"

    if p \* q < 32:

        return "p \* q < 32"

    if math.isnan(ds):

        return "ds is NaN"

    if not e:

        set\_autocomplete\_e(coprime(fi(p \* q)))

        return "set the parameter e"

    return set\_d(fi(p \* q), e)

def RSA\_DS\_encrypt(open\_text, p, q, e, alph):

    encoded\_text = ""

    for char in open\_text:

        encoded\_char = encodingFormat(char)  *# Применяем encodingFormat() к каждому символу*

        if encoded\_char:  *# Проверяем, что символ был успешно закодирован*

            encoded\_text += encoded\_char

        else:

            return "Входной текст содержит символы, которые не могут быть закодированы"

    n = p \* q

    h = hash(encoded\_text, n, alph)

*# Проверка взаимной простоты e и f*

    f = (p - 1) \* (q - 1)

    if gcd(e, f) != 1:

        return "Невозможно вычислить D: e и f не взаимно просты"

*# Вычисление D так, чтобы E \* D = 1 mod f*

    d = mod\_inverse(e, f)

    return str((h \*\* comparison([e, 1, fi(n)])) % n), d

def RSA\_DS\_decrypt(open\_text, p, q, e, ds, alph):

    encoded\_text = ""

    for letter in open\_text:

        encoded\_char = encodingFormat(letter)  *# Применяем encodingFormat() к каждому символу*

        if encoded\_char:  *# Проверяем, что символ был успешно закодирован*

            encoded\_text += encoded\_char

        else:

            return "Введёный текст содержит запрещённые символы"

    n = p \* q

    h = hash(encoded\_text, n, alph)

*# Проверка взаимной простоты e и f*

    f = (p - 1) \* (q - 1)

    if gcd(e, f) != 1:

        return "Невозможно вычислить D: e и f не взаимно просты"

*# Вычисление D так, чтобы E \* D = 1 mod f*

    d = mod\_inverse(e, f)

    decrypted\_hash = (ds \*\* e) % n

    return "Подпись верна" if decrypted\_hash == h else "Подпись верна", d

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

def mod\_inverse(a, m):

    g, x, y = extended\_gcd(a, m)

    if g != 1:

        raise Exception('Обратного элемента не существует')

    return x % m

def extended\_gcd(a, b):

    if a == 0:

        return b, 0, 1

    else:

        gcd, x, y = extended\_gcd(b % a, a)

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

def main():

    print("RSA (ЭЦП)")

    action = int(input("Выберите действие:\n 1) Подписать\n 2) Проверить\n"))

*# Создание подписи.*

    if (action == 1):

        open\_text = inputText()

*# Ввод параметров.*

        while True:

            p = int(input("Введите p(Простое): "))

            if isPrime(p):

                break

            else:

                print('Неверное p, оно должно быть простым')

        while True:

            q = int(input("Введите q(Простое): "))

            if isPrime(q):

                break

            else:

                print('Неверное q, оно должно быть простым')

        n = p \* q

        print('n =', n)

        while True:

            n = int(input("Введите n(n >= 32): "))

            if n >= 32:

                break

            else:

                print('Неверное n, оно должно быть больше или равно 32')

        f = (p-1)\*(q-1) *# ф-я Эйлера.*

        print("n, f: ", n, f)

        while True:

            e = int(input("Введите случайное целое число e, взаимно простое с f: "))

            if coprime(e, f):

                break

            else:

                print('Неверное e, оно должно быть взаимно простым с f и не равным d')

        print("Вычисление подписи...")

        signature = RSA\_DS\_encrypt(open\_text, p, q, e, alph)

        print('ЭЦП:', signature)

        saveOutput(str(signature))

        print("Подпись помещена в output.txt")

*# Проверка подписи.*

    elif (action == 2):

        open\_text = inputText()

        signature = input("Введите подпись для проверки: ")

        p = int(input("Введите простое число p: "))

        q = int(input("Введите простое число q: "))

        e = int(input("Введите значение e: "))

        ds = int(input("Введите значение ds: "))

        result = RSA\_DS\_decrypt(open\_text, p, q, e, ds, alph)

        print("Результат проверки:", result)

        saveOutput(result)

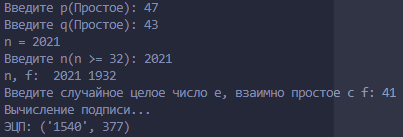
        print("Результат проверки подписи в output.txt")

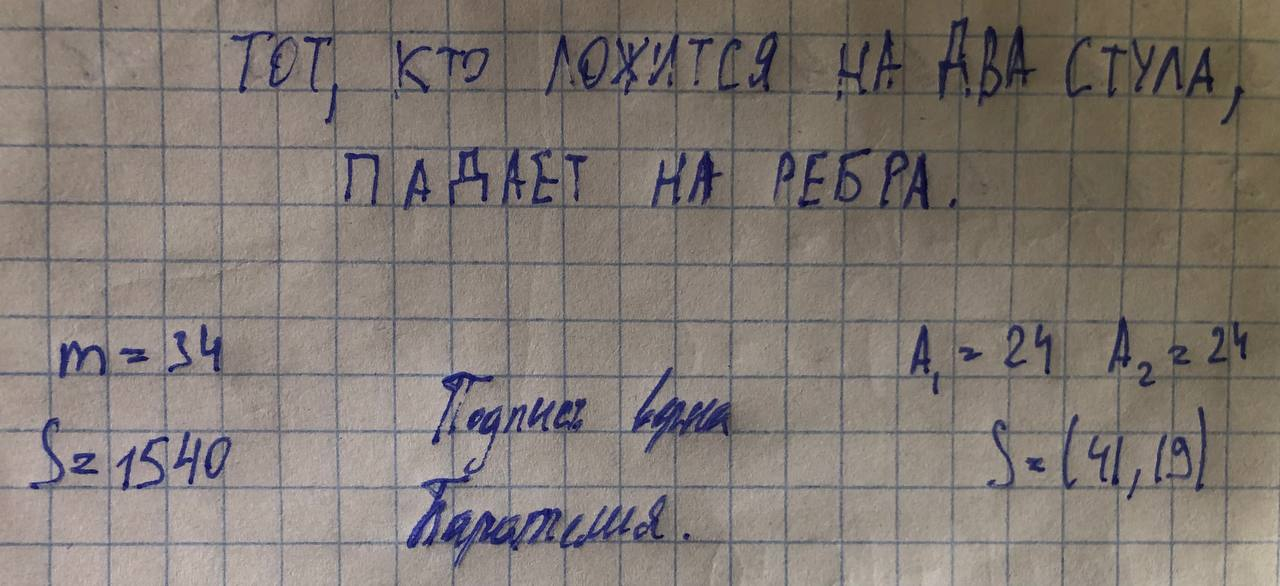
*# Некорректный ввод.*

    else:

        print("Некорректный ввод.")

**Тестирование**

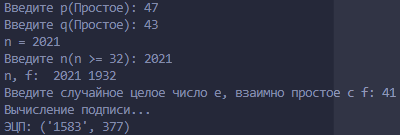
****



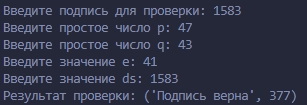
**Работа с текстом не менее 1000 знаков (шифрование и**

**расшифрование с указанием ключа)**

**Зашифрование**

****

**Расшифрование**

****