МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Схема подписи DSA**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Таранова Алексея Вадимовича

|  |  |  |
| --- | --- | --- |
| Преподаватель  аспирант | \_\_\_\_\_\_\_\_\_\_ | Р. А. Фарахутдинов |
|  | подпись, дата |  |

Саратов 2023

**1. Теоретическая часть**

Цель работы – изучение схемы ЭЦП – схемы подписи DSA и ее программная реализация.

Для генерации пары ключей выбираются два простых числа и , такие, что . Затем вычисляются , где – любое число, меньшее , для которого получится больше 1, – число меньше и . В алгоритме также используется однонаправленная хэш-функция . Параметры открыты и могут быть общими для пользователей сети. Закрытый ключ – , открытый ключ – . Чтобы подписать сообщение :

1. Алиса генерирует случайное .

2. Алиса генерирует , . Её подписью служат параметры и , которые она посылает Бобу.

3. Боб проверят подпись, вычисляя , , , . Если , то подпись правильная.

**2. Описание программы**

Функция main() предоставляет пользователю выбор между подписью сообщения (1) и проверкой подписи (2). В зависимости от выбора вызываются соответствующие функции для выполнения соответствующих шагов протокола.

Функция protocol\_gen() отвечает за подпись сообщения. У пользователя каждый шаг спрашивается, выполнять ли шаг заново или использовать сохраненные ранее значения. Сначала генерируются простые числа p и q с помощью функции create\_p(). Затем генерируются ключи (create\_keys()).

После чего начинается выполнение шагов 1 и 2 (first\_step() и second\_step()) если пользователь выбрал вариант подписать сообщение: Алиса генерирует случайное k<q, а затем генерирует , . Её подписью служат параметры и , которые она посылает Бобу.

Если же был выбран вариант проверки подписи, то запускается protocol\_check(), а в нём шаг 3, реализованный в функции third\_step(): Боб проверят подпись, вычисляя , , , . Если , то подпись правильная.

Для вычисления обратных чисел используется расширенный алгоритм Евклида (extended\_euclid(x1, x2)).

**3. Пример запуска программы**

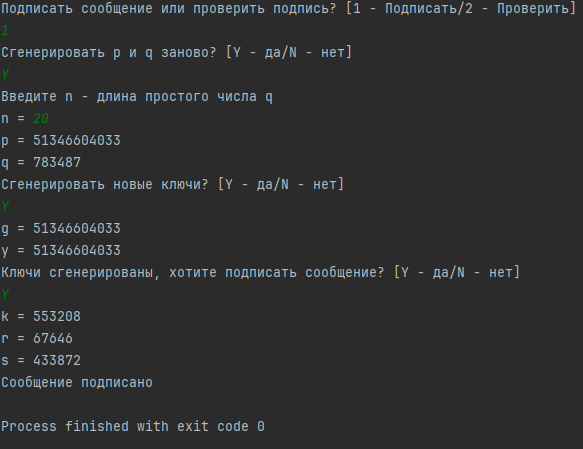
****

Рисунок 1 – Подпись сообщения

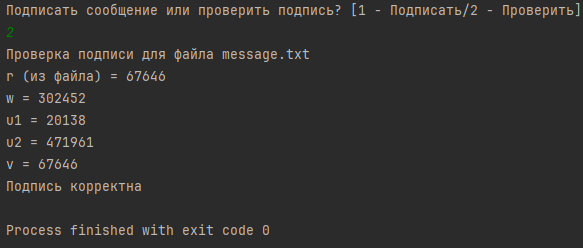


Рисунок 2 – Проверка подписи

**ПРИЛОЖЕНИЕ А**

Листинг 1 – main.py

import random

import hashlib

from sympy import isprime

def extended\_euclid(x1, x2):

o\_r, r = x1, x2

o\_s, s = 1, 0

o\_t, t = 0, 1

while r != 0:

q = o\_r // r

o\_r, r = r, o\_r - q \* r

o\_s, s = s, o\_s - q \* s

o\_t, t = t, o\_t - q \* t

return o\_r, o\_s, o\_t

def fast\_exponentiation(base, exponent, mod):

result = 1

base = base % mod

while exponent > 0:

if exponent % 2 == 1:

result = (result \* base) % mod

exponent = exponent // 2

base = (base \* base) % mod

return result

def create\_p():

print("Введите n - длина простого числа q\nn = ", end='')

exp = int(input())

q = random.randint(pow(2, exp - 1), pow(2, exp))

while not isprime(q):

q = random.randint(pow(2, exp - 1), pow(2, exp))

n = 2

while not isprime(q \* (2 \*\* n) + 1):

n += 1

p = q \* (2 \*\* n) + 1

f = open('p\_q.txt', 'w')

f.write(str(p) + ' ' + str(q))

f.close()

print("p =", p)

print("q =", q)

return 0

def create\_keys():

f = open('p\_q.txt', 'r')

p, q = map(int, f.read().split())

f.close()

while True:

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

g = fast\_exponentiation(h, (p - 1) // q, p)

if g > 1:

break

x = random.randint(2, q - 1)

y = fast\_exponentiation(g, x, p)

f = open('close\_keys.txt', 'w')

f.write(str(x))

f.close()

f = open('open\_keys.txt', 'w')

f.write(str(p) + ' ' + str(q) + ' ' + str(g) + ' ' + str(y))

f.close()

print("g =", p)

print("y =", p)

return 0

def first\_step():

f = open('open\_keys.txt', 'r')

\_, q, \_, \_ = map(int, f.read().split())

f.close()

k = random.randint(2, q - 1)

f = open('k.txt', 'w')

f.write(str(k))

f.close()

print("k =", k)

return 0

def second\_step():

f = open('k.txt', 'r')

k = int(f.read())

f.close()

f = open('open\_keys.txt', 'r')

p, q, g, \_ = map(int, f.read().split())

f.close()

f = open('close\_keys.txt', 'r')

x = int(f.read())

f.close()

r = fast\_exponentiation(g, k, p) % q

\_, rev\_k, \_ = extended\_euclid(k, q)

f = open('message.txt', 'r', encoding='utf-8')

m = f.read()

f.close()

s = (rev\_k \* (int(hashlib.sha1(bytes(m, encoding='utf-8')).hexdigest(), 16) + x \* r)) % q

f = open('sign.txt', 'w')

f.write(str(r) + ' ' + str(s))

f.close()

print("r =", r)

print("s =", s)

return 0

def third\_step():

f = open('open\_keys.txt', 'r')

p, q, g, y = map(int, f.read().split())

f.close()

f = open('sign.txt', 'r')

r, s = map(int, f.read().split())

print("r (из файла) =", r)

\_, w, \_ = extended\_euclid(s, q)

f = open('message.txt', 'r', encoding='utf-8')

m = f.read()

f.close()

u1 = (int(hashlib.sha1(bytes(m, encoding='utf-8')).hexdigest(), 16) \* w) % q

u2 = (r \* w) % q

v = (fast\_exponentiation(g, u1, p) \* fast\_exponentiation(y, u2, p) % p) % q

print("w =", w)

print("u1 =", u1)

print("u2 =", u2)

print("v =", v)

return v == r

def protocol\_gen():

print("Сгенерировать p и q заново? [Y - да/N - нет]")

while True:

f = input()

if f == 'Y':

create\_p()

break

elif f == 'N':

break

else:

print("Некорректный ввод")

print("Сгенерировать новые ключи? [Y - да/N - нет]")

while True:

f = input()

if f == 'Y':

create\_keys()

print("Ключи сгенерированы, хотите подписать сообщение? [Y - да/N - нет]")

break

elif f == 'N':

print("Хотите подписать сообщение? [Y - да/N - нет]")

break

else:

print("Некорректный ввод")

while True:

f = input()

if f == 'Y':

first\_step()

break

elif f == 'N':

return 0

else:

print("Некорректный ввод")

second\_step()

print("Сообщение подписано")

def protocol\_check():

print("Проверка подписи для файла message.txt")

if third\_step():

print("Подпись корректна")

else:

print("Подпись некорректна")

def main():

print("Подписать сообщение или проверить подпись? [1 - Подписать/2 - Проверить]")

while True:

f = input()

if f == '1':

protocol\_gen()

return 0

elif f == '2':

protocol\_check()

return 0

else:

print("Некорректный ввод")

main()