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

Мызникова Сергея Анатольевича

|  |  |  |
| --- | --- | --- |
| Преподаватель, доцент | \_\_\_\_\_\_\_\_\_\_ | В.Е. Новиков |
|  | подпись, дата |  |

Саратов 2024

**1 Постановка задачи**

Цель работы: изучить алгоритм схема подписи DSA и привести его программную реализацию

**2 Теоретические сведения**

DSA включает в себя два алгоритма (S, V): для создания подписи сообщения (S) и для ее проверки (V).

Оба алгоритма вначале вычисляют хеш сообщения, используя криптографическую хеш-функцию. Алгоритм S использует хеш и секретный ключ для создания подписи, алгоритм V использует хеш сообщения, подпись и открытый ключ для проверки подписи.

Стоит подчеркнуть, что фактически подписывается не сообщение (произвольной длины), а его хеш, поэтому неизбежны коллизии и одна подпись, вообще говоря, действительна для нескольких сообщений с одинаковым хешем. Поэтому выбор достаточно "хорошей" хеш-функции очень важен для всей системы в целом. В первой версии стандарта использовалась хеш-функция SHA-1, в последней версии также можно использовать любой алгоритм семейства SHA-2. В августе 2015 был опубликован FIPS-202, описывающий новую хеш-функцию SHA-3. Но на сегодняшний день она не включена в стандарт DSS.

**Алгоритм**

**Генерация параметров и ключей**

1. Выбор криптографической хеш-функции *H(x)*
2. Выбор простого числа *q*, размерность которого *N* в битах совпадает с размерностью в битах значений хеш-функции *H(x)*.
3. Выбор простого числа p, такого, что (*p* - 1) делится на *q*.
4. Выбор числа *g* (*g* != 1) такого, что его мультипликативный порядок по модулю *p* равен *q*. Для его вычисления можно воспользоваться формулой
5. Секретный ключ представляет собой число *x* (0, *q*)
6. Открытый ключ вычисляется по формуле *y = gx mod p*

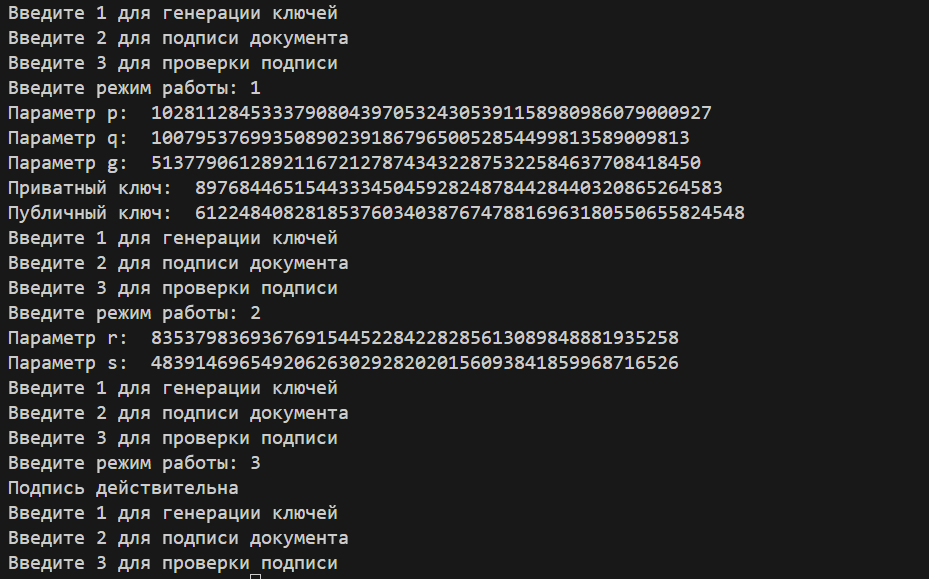
**Подпись сообщения**

1. Выбор случайного числа *k* (0, *q*)
2. Вычисление *r = (gk mod p) mod q*
3. Выбор другого *k*, если *r* = 0
4. Вычисление *s = k-1(H(m) + x \* r) mod q*
5. Выбор другого *k*, если *s* = 0
6. Подписью является пара *(r, s)*

**Проверка подписи**

1. Вычисление *w = s-1 mod q*
2. Вычисление *u1 = H(m) \* w mod q*
3. Вычисление *u2 = r \* w mod q*
4. Вычисление *v* = (*mod p) mod q*
5. Подпись верна, если *v* = *r*

**3 Тестирование программы**



**ПРИЛОЖЕНИЕ А**

import random

import hashlib

import sympy

def mod\_inverse(a, q):

    return pow(a, -1, q)

def gen\_param(bits):

    q = random.getrandbits(bits)

    while not sympy.isprime(q):

        q = random.getrandbits(bits)

    i = 2

    while True:

        p = q \* i + 1

        if sympy.isprime(p):

            break

        i += 1

    g = 1

    while g == 1:

        h = random.randint(2, p - 1)

        g = pow(h, (p - 1) // q, p)  # g = h^((p-1)/q) mod p

    return p, q, g

def generate\_keys(p, q, g):

    private = random.randint(1, q - 1)

    public = pow(g, private, p)

    return private, public

def sign\_message(p, q, g, x, message):

    k = random.randint(1, q - 1)

    r = pow(g, k, p) % q

    while r == 0:

        k = random.randint(1, q - 1)

        r = pow(g, k, p) % q

    k\_inv = mod\_inverse(k, q)

    hash\_m = hashlib.sha1()

    hash\_m.update(message.encode('utf-8'))

    hash\_m = int(hash\_m.hexdigest(), 16)

    s = (k\_inv \* (hash\_m + x \* r)) % q

    if s == 0:

        return sign\_message(p, q, g, x, message)  # Повторим, если s=0

    return r, s

def verify\_signature(p, q, g, y, message, r, s):

    if not (0 < r < q and 0 < s < q):

        return False

    hash\_m = hashlib.sha1()

    hash\_m.update(message.encode('utf-8'))

    hash\_m = int(hash\_m.hexdigest(), 16)

    w = mod\_inverse(s, q)

    u1 = (hash\_m \* w) % q

    u2 = (r \* w) % q

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

    return v == r

def write\_to\_file(files, message):

    with open(files, 'w') as file:

        file.write(','.join(map(str, message)))

def open\_file(files):

    with open(files, 'r') as file:

        data = tuple(map(str, file.read().split(',')))

    return data

def main():

    print("Введите 1 для генерации ключей")

    print("Введите 2 для подписи документа")

    print("Введите 3 для проверки подписи")

    k = int(input("Введите режим работы: "))

    try:

        if k == 1:

            with open('input.txt', 'r', encoding='utf-8') as file:

                message = file.read()

            hash\_m = hashlib.sha1()

            hash\_m.update(message.encode('utf-8'))

            hash\_m = int(hash\_m.hexdigest(), 16)

            tmp = bin(hash\_m)

            p, q, g = gen\_param(len(tmp) - 2)

            print("Параметр p: ", p)

            print("Параметр q: ", q)

            print("Параметр g: ", g)

            write\_to\_file("parametrs.txt", (p, q, g))

            private, public = generate\_keys(p, q, g)

            print("Приватный ключ: ", private)

            print("Публичный ключ: ", public)

            with open("private\_key.txt", 'w') as file:

                file.write(str(private))

            with open("public\_key.txt", 'w') as file:

                file.write(str(public))

        elif k == 2:

            p, q, g = open\_file("parametrs.txt")

            private = open\_file("private\_key.txt")[0]

            with open('input.txt', 'r', encoding='utf-8') as file:

                message = file.read()

            r, s = sign\_message(int(p), int(q), int(g), int(private), message)

            print("Параметр r: ", r)

            print("Параметр s: ", s)

            write\_to\_file("signed\_message.txt", (r, s))

        elif k == 3:

            p, q, g = open\_file("parametrs.txt")

            public = open\_file("public\_key.txt")[0]

            r, s = open\_file("signed\_message.txt")

            with open('input.txt', 'r', encoding='utf-8') as file:

                message = file.read()

            if verify\_signature(int(p), int(q), int(g), int(public), message, int(r), int(s)):

                print("Подпись действительна")

                return

            else:

                print("Подпись недействительна")

                return

    except Exception as e:

        print("Error: ", e)

while True:

    main()