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

Норикова Павла Сергеевича

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

Саратов 2023

**ВВЕДЕНИЕ**

Цель работы – изучение и реализация схемы подписи DSA.

**1 Теория**

DSA (англ. Digital Signature Algorithm – алгоритм цифровой подписи) – криптографический алгоритм с использованием закрытого ключа (из пары ключей: <открытый; закрытый>) для создания электронной подписи, но не для шифрования (в отличие от RSA и схемы Эль-Гамаля). Подпись создается секретно (закрытым ключом), но может быть публично проверена (открытым ключом). Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность. Алгоритм основан на вычислительной сложности взятия логарифмов в конечных полях.

* 1. **Параметры схемы цифровой подписи**

1. Выбирается простое число .
2. Выбор криптографической хеш-функции .
3. Выбор простого числа , размерность которого в битах совпадает с размерностью в битах значений хеш-функции .
4. Выбор простого числа , такого, что делится на .
5. Выбор числа такого, что его мультипликативный порядок по модулю равен . Для его вычисления можно воспользоваться формулой  
   где — некоторое произвольное число.
   1. **Генерация ключей**
6. Секретный ключ представляет собой число
7. Открытый ключ вычисляется по формуле
   1. **Подпись сообщения**
8. Выбор случайного числа
9. Вычисление
10. Выбор другого , если
11. Вычисление
12. Выбор другого , если
13. Подписью является пара общей длины
    1. **Проверка подписи**
14. Вычисление
15. Вычисление
16. Вычисление
17. Вычисление
18. Подпись верна, если

**2 Практическая реализация**

**2.1 Описание программы**

Функции *gen\_p*, *gen\_q* и *gen\_g* генерируют числа *p*, *q* и *g* соответственно.

Функции *gcd\_ex* и *gcd* реализуют расширенный и обычный алгоритм Евклида соответственно.

На вход программе подается сообщение .

**2.2 Тестирование программы**

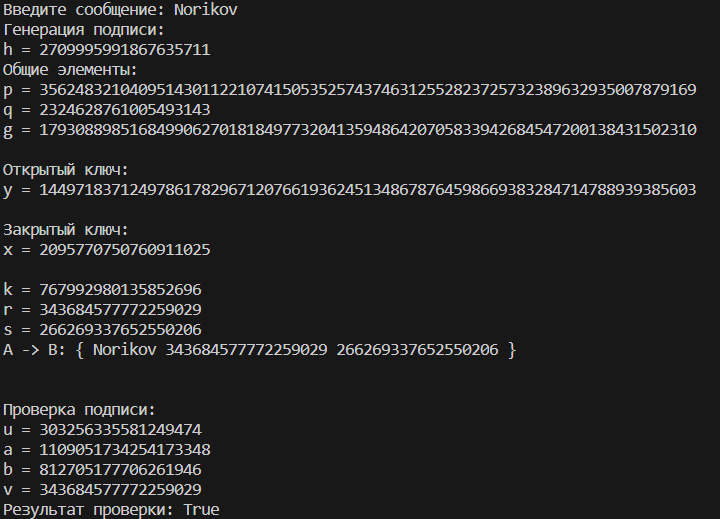


Рисунок 1 - Работа протокола

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

import random

from sympy import isprime

def gcd(a, b):

while b:

a, b = b, a % b

return a

def gcd\_ex(a, b):

if b == 0:

return a, 1, 0

else:

gcd, x1, y1 = gcd\_ex(b, a % b)

x = y1

y = x1 - (a // b) \* y1

return gcd, x, y

def gen\_p(q):

res = 2 \* q

while not isprime(res + 1):

res \*= 2

return res + 1

def gen\_q(L):

res = "0"

while not isprime(int(res, 2)):

res = ""

for i in range (1, L - 1):

random.seed()

res += str(random.randint(0,100)%2)

res = '1' + res + '1'

return int(res, 2)

def gen\_g(q, p):

res = 1

h = 2

while res == 1:

res = pow(h, (p - 1) // q, p)

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

return res

def main(m):

print('Генерация подписи:')

#h = m

h = abs(int(hash(str(m))))

print('h =', h)

q = gen\_q(len(bin(h)) - 2)

p = gen\_p(q)

g = gen\_g(q, p)

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

y = pow(g, x, p)

print('Общие элементы:', '\np =', p, '\nq =', q, '\ng =', g, '\n')

print('Открытый ключ:', '\ny =', y, '\n')

print('Закрытый ключ:', '\nx =', x, '\n')

s = 0

k = 0

while s == 0 or r == 0:

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

#k = 3

r = pow(g, k, p) % q

s = (gcd\_ex(k, q)[1] \* (h + x\*r)) % q

#print(s)

print('k =', k)

print('r =', r)

print('s =', s)

print('A -> B: {', m, r, s, '}\n')

print('\nПроверка подписи:')

u = gcd\_ex(s, q)[1]

print('u =', u)

a = (h \* u) % q

print('a =', a)

b = (r \* u) % q

print('b =', b)

v = (pow(g, a, p) \* pow(y, b, p)) % p % q

print('v =', v)

print('Результат проверки:', v == r)

s = str(input('Введите сообщение: '))

main(s)