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

**Протокол аутентификации Шнорра**

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 протокола аутентификации Шнорра.

**1 Теория**

Схема Шнорра – одна из наиболее эффективных и теоретически обоснованных схем аутентификации. Безопасность схемы основывается на трудности вычисления дискретных логарифмов. Предложенная Клаусом Шнорром схема является модификацией схем Эль-Гамаля и Фиата-Шамира, но имеет меньший размер подписи.

* 1. **Генерация ключей**

1. Выбирается простое число .
2. Выбирается другое простое число , такое, что .
3. Выбирается число , отличное от 1, такое, что .
4. Пегги выбирает случайное целое число меньшее .
5. Пегги вычисляет .
6. Общедоступный ключ Пегги – , секретный ключ Пегги – .
   1. **Проверка подлинности**
7. Алиса выбирает случайное число , меньшее , и вычисляет . Эти вычисления являются предварительными и могут быть выполнены задолго до появления Боба.
8. Алиса посылает Бобу.
9. Боб выбирает случайное число из диапазона от 0 до и отправляет его Алисе.
10. Алиса вычисляет и посылает Бобу.
11. Боб проверяет что .

**2 Практическая реализация**

**2.1 Описание программы**

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

Функция *gen\_key* генерирует открытый и закрытый ключ.

На вход программе подается битовые длины чисел *p* и *q*, а также параметр надежности *t*.

**2.2 Тестирование программы**

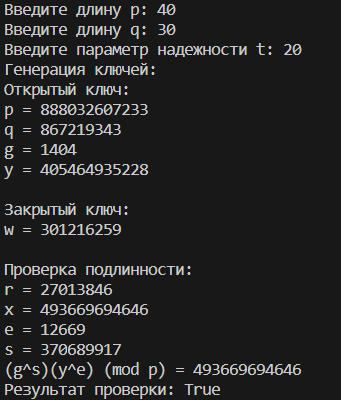


Рисунок 1 - Работа протокола

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

import random

from sympy import isprime

def gen\_p(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\_p2(L, q):

res = "0"

while (not isprime(int(res, 2))) or (int(res, 2) - 1 % q != 0):

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\_p3(L, L2, q):

iter = pow(2, L - L2)

res = iter \* q

while not isprime(res + 1):

res += q

return res + 1

def gen\_q(L, p):

res = "0"

while (not isprime(int(res, 2))) or (p - 1 % int(res, 2) != 0) :

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\_q2(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, L\_p):

L = 2

while True:

L += 1

print(L)

res = "0"

it = 0

while pow(int(res, 2), q, p) != 1 and it < 1000:

res = ""

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

random.seed()

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

res = '1' + res + '1'

it += 1

if it < 1000:

break

return int(res, 2)

def gen\_g2(q, p):

res = 2

while pow(res, q, p) != 1:

res += 1

#print(res)

return res

def gen\_key2(L\_p, L\_q):

q = gen\_q2(L\_q)

p = gen\_p3(L\_p, L\_q, q)

g = gen\_g2(q, p)

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

y = pow(g, q - w, p)

print('Открытый ключ:', '\np =', p, '\nq =', q, '\ng =', g, '\ny =', y, '\n')

print('Закрытый ключ:', '\nw =', w, '\n')

return (p, q, g, y, w)

def main(L\_p, L\_q, t):

print('Генерация ключей:')

p, q, g, y, w = gen\_key2(L\_p, L\_q)

print('Проверка подлинности:')

r = random.randint(1, q)

print('r =', r)

x = pow(g, r, p)

print('x =', x)

e = random.randint(0, pow(2, t) - 1)

print('e =', e)

s = (r + w\*e) % q

print('s =', s)

x2 = pow(g, s, p) \* pow(y, e, p) % p

print('(g^s)(y^e) (mod p) =', x2)

print('Результат проверки:', x == x2)

L\_p = int(input('Введите длину p: '))

L\_q = int(input('Введите длину q: '))

t = int(input('Введите параметр надежности t: '))

main(L\_p, L\_q, t)