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.

1.1 Генерация ключей

1. Выбирается простое число p.
2. Выбирается другое простое число q, такое, что $p - 1 \equiv 1 \ (mod\ q)$.
3. Выбирается число g, отличное от 1, такое, что $g^q \equiv 1 \ (mod\ p)$.
4. Пегги выбирает случайное целое число w меньшее q.
5. Пегги вычисляет $y = g^{q-w} \ (mod\ p)$.
6. Общедоступный ключ Пегги – (p, q, g, y), секретный ключ Пегги – w.

1.2 Проверка подлинности

1. Алиса выбирает случайное число r, меньшее q, и вычисляет $x = g^r \ (mod\ p)$. Эти вычисления являются предварительными и могут быть выполнены задолго до появления Боба.
2. Алиса посылает x Бобу.
3. Боб выбирает случайное число e из диапазона от 0 до $2^t - 1$ и отправляет его Алисе.
4. Алиса вычисляет $s = r + we \ (mod\ q)$ и посылает s Бобу.
5. Боб проверяет что $x = g^s y^e (mod\ p)$.

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

```
Введите длину p: 40
Введите длину q: 30
Введите параметр надежности t: 20
Генерация ключей:
Открытый ключ:
p = 888032607233
q = 867219343
g = 1404
y = 405464935228

Закрытый ключ:
w = 301216259

Проверка подлинности:
r = 27013846
x = 493669694646
e = 12669
s = 370689917
(g^s)(y^e) (mod p) = 493669694646
Результат проверки: True
```

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)
```