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

**Алгоритм Хьюза**

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 открытого распределения ключей Hughes.

**1 Теория**

Алгоритм Хьюза является вариантом алгоритма Диффи — Хеллмана. Он позволяет Алисе генерировать ключ и послать его Бобу.

Преимущество этого протокола над Diffie–Hellman состоит в том, что k можно вычислить заранее, до взаимодействия, и Алиса может шифровать сообщения с помощью k задолго до установления соединения с Бобом. Она может шифровать сообщения сразу множеству людей, а передать ключ позднее каждому по отдельности.

Протокол Ди́ффи — Хе́ллмана (англ. Diffie–Hellman key exchange protocol, DH) — криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования.

**1.1 Описание алгоритма**

Для начала выбираются или генерируются 2 числа: большое простое число n и порождающий элемент g.

1. Алиса выбирает большое целое число и вычисляет .
2. Боб выбирает большое целое число и посылает Алисе .
3. Алиса посылает Бобу .
4. Боб вычисляет и .

Если все выполнено правильно, .

**2 Практическая реализация**

**2.1 Описание программы**

Функции *gen\_p*, *gen\_g* и *gen\_y* генерируют числа *p*, *g* и *y* соответственно.

Функция *gcd\_ex* реализует расширенный алгоритм Евклида.

На вход программе подается битовая длина числа *p*.

**2.2 Тестирование программы**

Протестируем программу для битовой длины числа *p* равной 40.

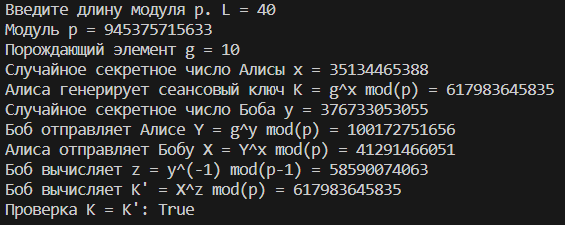


Рисунок 1 - Работа программы

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

import random

import math

from sympy import isprime

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(L):

res = "0"

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

res = '0'

while int(res, 2) % 4 != 1:

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(p):

fact = []

phi = p - 1

n = phi

for i in range(2, math.floor(math.sqrt(n) + 1)):

if(n % i == 0):

fact.append(i)

while(n % i == 0):

n /= i

if n > 1:

fact.append(n)

for res in range (2, p + 1):

ok = True

for i in range(len(fact)):

if not ok:

break

ok = ok and pow(res, int(phi // fact[i]), p) != 1

if ok:

return res

return -1

def gen\_y(p):

y = random.randint(2, p-1)

while math.gcd(y, p - 1) != 1:

y = random.randint(2, p-1)

return y

while 1:

L = int(input('Введите длину модуля p. L = '))

g = 0

p = gen\_p(L)

g = gen\_g(p)

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

K = pow(g, x, p)

print('Модуль p =', p)

print('Порождающий элемент g =', g)

print('Случайное секретное число Алисы x =', x)

print('Алиса генерирует сеансовый ключ K = g^x mod(p) =', K)

y = gen\_y(p)

print('Случайное секретное число Боба y =', y)

Y = pow(g, y, p)

print('Боб отправляет Алисе Y = g^y mod(p) =', Y)

X = pow(Y, x, p)

print('Алиса отправляет Бобу X = Y^x mod(p) =', X)

z = gcd\_ex(y, p - 1)[1]

K\_ = pow(X, z, p)

print('Боб вычисляет z = y^(-1) mod(p-1) =', z)

print('Боб вычисляет K\' = X^z mod(p) =', K\_)

print("Проверка K = K\':", K == K\_, '\n\n')