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 EKE на базе алгоритма Эль-Гамаля**

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 передачи секретного ключа по открытому каналу EKE на базе алгоритма Эль-Гамаля.

**1 Теория**

Реализация EKE на базе алгоритма Эль-Гамаля проста, можно даже упростить основной протокол. Пусть – модуль, а – порождающий элемент. и служат частями открытого ключа, общими для всех пользователей. Закрытым ключом является случайное число . Открытым – . На этапе 1 Алиса посылает Бобу следующее сообщение

Боб выбирает случайное число и на этапе 2 посылает Алисе следующее сообщение

* 1. **Описание алгоритма**

1. .
2. .
3. . Алиса расшифровывает сообщение, получая . Она генерирует случайную строку , шифрует ее с помощью и посылает Бобу.
4. . Боб расшифровывает , генерирует случайную строку шифрует обе строки с помощью и посылает Алисе.
5. . Алиса расшифровывает , сравнивает полученную с , сгенерированной на шаге 3. Если они совпали, то она шифрует с помощью и посылает Бобу.
6. Боб расшифровывает , сравнивает полученную с , сгенерированной на шаге 4. Если они совпали, то протокол завершен.

**2 Практическая реализация**

**2.1 Описание программы**

Функции *encr* и *decr* осуществляют шифрование и расшифрование соответственно.

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

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

Функция *gen\_str* генерирует случайную строку.

Функция *int\_to\_key* преобразует число в ключ шифрования.

На вход программе подается битовая длина числа *p*.

**2.2 Тестирование программы**

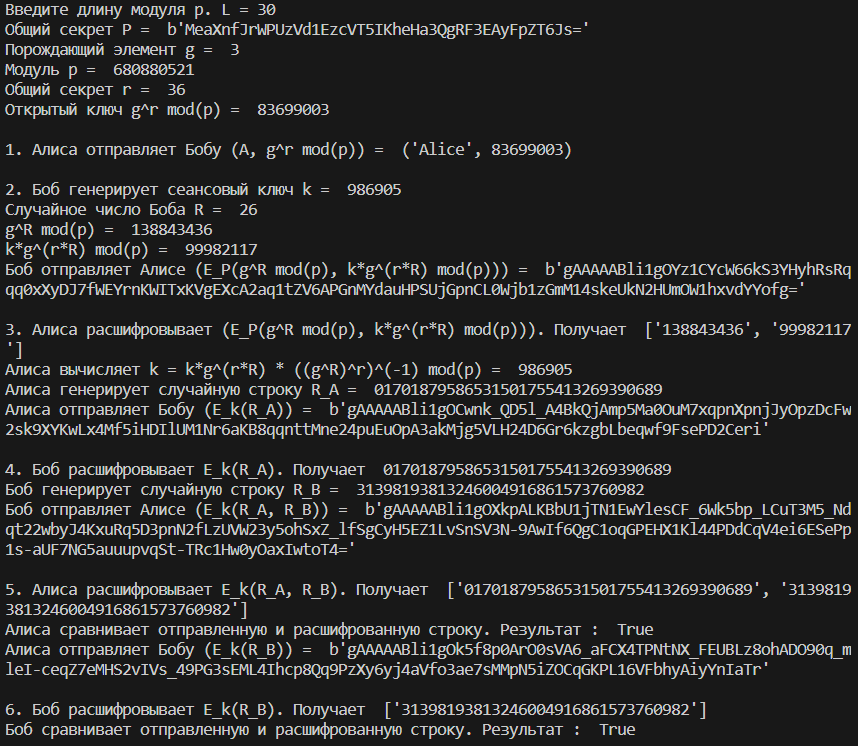


Рисунок 1 - Работа протокола

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

from cryptography.fernet import Fernet

from cryptography.hazmat.primitives.asymmetric import ec

from cryptography.hazmat.primitives import serialization

from cryptography.hazmat.primitives.asymmetric import utils

from cryptography.hazmat.backends import default\_backend

import base64

import os

from cryptography.fernet import Fernet

from cryptography.hazmat.primitives import hashes

from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

import time

import random

import math

def encr(key, x):

data = ''

for i in x:

if type(i) == bytes:

data += str(i, 'utf-8')

else:

data += str(i)

data += '\n'

data = data[:-1]

cipher\_suite = Fernet(key)

data = bytes(data, 'utf-8')

enc = cipher\_suite.encrypt(data)

return enc

def decr(key, enc):

cipher\_suite = Fernet(key)

dec = cipher\_suite.decrypt(enc)

dec = str(dec, 'utf-8')

res = dec.split('\n')

return res

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 J(a1, b1):

a = a1

b = b1

st\_b = b1

r = 1

while a != 0:

t = 0

while a % 2 == 0:

t += 1

a = int(a//2)

if t % 2 == 1:

if b % 8 == 3 or b % 8 == 5:

r = -r

if a % 4 == b % 4 == 3:

r = -r

c = a

a = b%c

b = c

if r == -1:

return st\_b - 1

return r

def sol(n):

if n == 0 or n == 1 or n % 2 == 0:

return False

for i in range(0,10):

a = random.randint(1, n-1)

if a%2 == 0:

a += 1

if math.gcd(a, n) > 1:

return False

if pow(a, int((n-1)//2), n) != J(a, n):

return False

return True

def gen\_p(L):

res = "0"

while not sol(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\_str():

x = ''

for i in range(32):

x += str(random.randint(0,9))

#print(x[2:])

return x

def int\_to\_key(x):

password = bytes(str(x), 'utf-8')

kdf = PBKDF2HMAC(

algorithm = hashes.SHA256(),

length = 32,

salt = bytes(str(x), 'utf-8'),

iterations = 480000,

)

key = base64.urlsafe\_b64encode(kdf.derive(password))

return key

while 1:

####################0####################

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

g = 0

p = gen\_p(L)

g = gen\_g(p)

P = Fernet.generate\_key()

print('Общий секрет P = ', P)

r = random.randint(10, 100)

grp = pow(g, r, p)

print('Порождающий элемент g = ', g)

print('Модуль p = ', p)

print('Общий секрет r = ', r)

print('Открытый ключ g^r mod(p) = ', grp, '\n')

####################1####################

A = 'Alice'

s1 = (A, grp)

print('1. Алиса отправляет Бобу (А, g^r mod(p)) = ', s1, '\n')

####################2####################

K = random.randint(10, 1000000) % p

K\_key = int\_to\_key(K)

#print(111, K\_key)

print('2. Боб генерирует сеансовый ключ k = ', K)

R = random.randint(10, 100)

print('Случайное число Боба R = ', R)

s2\_1 = pow(g, R, p)

s2\_2 = K \* pow(g, r \* R, p) % p

s2 = encr(P, (str(s2\_1), str(s2\_2)))

print('g^R mod(p) = ', s2\_1)

print('k\*g^(r\*R) mod(p) = ', s2\_2)

print('Боб отправляет Алисе (E\_P(g^R mod(p), k\*g^(r\*R) mod(p))) = ', s2, '\n')

####################3####################

s2\_decr = decr(P, s2)

print('3. Алиса расшифровывает (E\_P(g^R mod(p), k\*g^(r\*R) mod(p))). Получает ', s2\_decr)

g\_rR = pow(int(s2\_decr[0]), r, p)

obr\_g\_rR = gcd\_ex(g\_rR, p)[1]

A\_K = obr\_g\_rR \* int(s2\_decr[1]) % p

print('Алиса вычисляет k = k\*g^(r\*R) \* ((g^R)^r)^(-1) mod(p) = ', A\_K)

A\_K = int\_to\_key(A\_K)

R\_A = gen\_str()

print('Алиса генерирует случайную строку R\_A = ', R\_A)

s3 = encr(A\_K, {R\_A})

print('Алиса отправляет Бобу (E\_k(R\_A)) = ', s3, '\n')

####################4####################

s3\_decr = decr(K\_key, s3)

print('4. Боб расшифровывает E\_k(R\_A). Получает ', s3\_decr[0])

R\_B = gen\_str()

print('Боб генерирует случайную строку R\_B = ', R\_B)

s4 = encr(K\_key, (s3\_decr[0], R\_B))

print('Боб отправляет Алисе (E\_k(R\_A, R\_B)) = ', s4, '\n')

####################5####################

s4\_decr = decr(A\_K, s4)

print('5. Алиса расшифровывает E\_k(R\_A, R\_B). Получает ', s4\_decr)

print('Алиса сравнивает отправленную и расшифрованную строку. Результат : ', s4\_decr[0] == R\_A)

s5 = encr(A\_K, {R\_B})

print('Алиса отправляет Бобу (E\_k(R\_B)) = ', s5, '\n')

####################6####################

s5\_decr = decr(K\_key, s5)

print('6. Боб расшифровывает E\_k(R\_B). Получает ', s5\_decr)

print('Боб сравнивает отправленную и расшифрованную строку. Результат : ', s5\_decr[0] == R\_B, '\n\n\n')