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 $p$ – модуль, а $g$ – порождающий элемент. $p$ и $g$ служат частями открытого ключа, общими для всех пользователей. Закрытым ключом является случайное число $r$. Открытым – $g^r \ mod \ p$. На этапе 1 Алиса посылает Бобу следующее сообщение

$$\text{Алиса}, g^r \ mod \ p$$

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

$$E_P(g^R \ mod \ p, Kg^{rR} \ mod \ p)$$

### 1.1 Описание алгоритма

1. $Alice \rightarrow \{A, g^r \ mod(p)\} \rightarrow Bob.$
2. $Bob \rightarrow \{E_P(g^R \ mod(p), kg^{rR} \ mod(p))\} \rightarrow Alice.$
3. $Alice \rightarrow \{E_k(R_A)\} \rightarrow Bob.$ Алиса расшифровывает сообщение, получая $k$. Она генерирует случайную строку $R_A$, шифрует ее с помощью $k$ и посылает Бобу.
4. $Bob \rightarrow \{E_k(R_A, R_B)\} \rightarrow Alice.$ Боб расшифровывает $E_k(R_A)$, генерирует случайную строку $R_B$ шифрует обе строки с помощью $k$ и посылает Алисе.
5. $Alice \rightarrow \{E_K(R_B)\} \rightarrow Bob.$ Алиса расшифровывает $E_k(R_A, R_B)$, сравнивает полученную $R_A$ с $R_A$, сгенерированной на шаге 3. Если они совпали, то она шифрует $R_B$ с помощью $k$ и посылает Бобу.
6. Боб расшифровывает $E_k(R_B)$, сравнивает полученную $R_B$ с $R_B$, 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

```
Введите длину модуля p. L = 30
Общий секрет P =  b'MeaXnfJrWPUzVd1EzcVT5IKheHa3QgRF3EAyFpZT6Js='
Порождающий элемент g =  3
Модуль p =  680880521
Общий секрет r =  36
Открытый ключ g^r mod(p) =  83699003

1. Алиса отправляет Бобу (A, g^r mod(p)) =  ('Alice', 83699003)

2. Боб генерирует сеансовый ключ k =  986905
Случайное число Боба R =  26
g^R mod(p) =  138843436
k*g^(r*R) mod(p) =  99982117
Боб отправляет Алисе (E_P(g^R mod(p), k*g^(r*R) mod(p))) =  b'gAAAAABli1gOYz1CYcW66kS3YHyhRsRq
qq0xXyDJ7fWEYrnKWITxKVgEXcA2aq1tZV6APGnMYdauHPSUjGpnCL0Wjb1zGmM14skeUkN2HUmOW1hxvdYYofg='

3. Алиса расшифровывает (E_P(g^R mod(p), k*g^(r*R) mod(p))). Получает  ['138843436', '99982117
']
Алиса вычисляет k = k*g^(r*R) * ((g^R)^r)^(-1) mod(p) =  986905
Алиса генерирует случайную строку R_A =  017018795865315017554132693906 89
Алиса отправляет Бобу (E_k(R_A)) =  b'gAAAAABli1gOCwnk_QD5l_A4BkQjAmp5Ma0OuM7xqpnXpnjJyOpzDcFw
2sk9XYKwLx4Mf5iHDIlUM1Nr6aKB8qqnttMne24puEuOpA3akMjg5VLH24D6Gr6kzgbLbeqwf9FsePD2Ceri'

4. Боб расшифровывает E_k(R_A). Получает  017018795865315017554132693906 89
Боб генерирует случайную строку R_B =  313981938132460049168615737609 82
Боб отправляет Алисе (E_k(R_A, R_B)) =  b'gAAAAABli1gOXkpALKBbU1jTN1EwYlesCF_6Wk5bp_LCuT3M5_Nd
qt22wbyJ4KxuRq5D3pnN2fLzUVW23y5ohSxZ_lfSgCyH5EZ1LvSnSV3N-9AwIf6QgC1oqGPEHX1Kl44PDdCqV4ei6ESePp
1s-aUF7NG5auuupvqSt-TRc1Hw0yOaxIwtoT4='

5. Алиса расшифровывает E_k(R_A, R_B). Получает  ['017018795865315017554132693906 89', '3139819
381324600491686157376098 2']
Алиса сравнивает отправленную и расшифрованную строку. Результат :  True
Алиса отправляет Бобу (E_k(R_B)) =  b'gAAAAABli1gOk5f8p0ArO0sVA6_aFCX4TPNtNX_FEUBLz8ohADO90q_m
leI-ceqZ7eMHS2vIVs_49PG3sEML4Ihcp8Qq9PzXy6yj4aVfo3ae7sMMpN5iZOCqGKPL16VFbhyAiyYnIaTr'

6. Боб расшифровывает E_k(R_B). Получает  ['313981938132460049168615737609 82']
Боб сравнивает отправленную и расшифрованную строку. Результат :  True
```

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 (A, 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')
```