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 анонимности Нурми-Саломаа-Сантина**

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 анонимности Нурми-Саломаа-Сантина.

**1 Теория**

Протоколы электронного голосования – это протоколы обмена данными для безопасного голосования через электронные технические средства.

Многие страны уже внедряют данный тип голосования и для уверенности в целостности, конфиденциальности и доступности выборов используют протоколы с доказанной защищенностью, которые должны реализовывать обязательные требования в том, что только голосующий может знать свой выбор, что проголосовать можно один раз и только участником, допущенным к выборам и решение проголосовавшего не может быть кем-либо изменено.

* 1. **Описание алгоритма**

Обозначения: – валидатор, – избиратель, – агентство, – секретные опознавательные метки.

1. отправляет всем до голосования.
2. отправляет весь набор , но без информации о том, кому они принадлежат.
3. создает свои ключи , и выкладывает в общий доступ , а также создает секретный ключ (), который нужен, чтобы никто не узнал содержимое бюллетеня до нужного момента.
4. формирует сообщение , где выражает свой выбор, подписывает , прикладывает к нему полученную и шифрует .
5. К зашифрованному тексту прикладывает и отправляет .
6. получает зашифрованный текст, по определяет, что он пришел от , но не знает от кого именно и как проголосовал, после публикует его.
7. Опубликованный зашифрованный текст служит информацией, чтобы отправил .
8. собирает ключи, расшифровывает текст, подсчитывает голоса и присоединяет к опубликованному зашифрованному тексту без .

**2 Практическая реализация**

**2.1 Описание программы**

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

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

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

Функция *V1* генерирует опознавательные метки.

Класс *B* включает в себя все необходимые данные и ключи избирателя, а также метод *B1*, реализующий процесс голосования.

На вход программе подается количество избирателей и кандидатов.

**2.2 Тестирование программы**

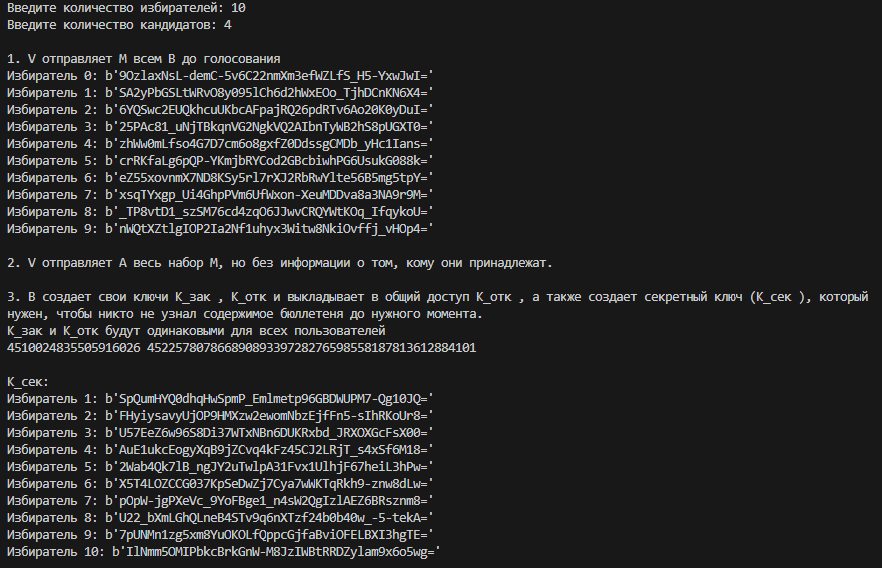


Рисунок 1 – Шаги 1-3

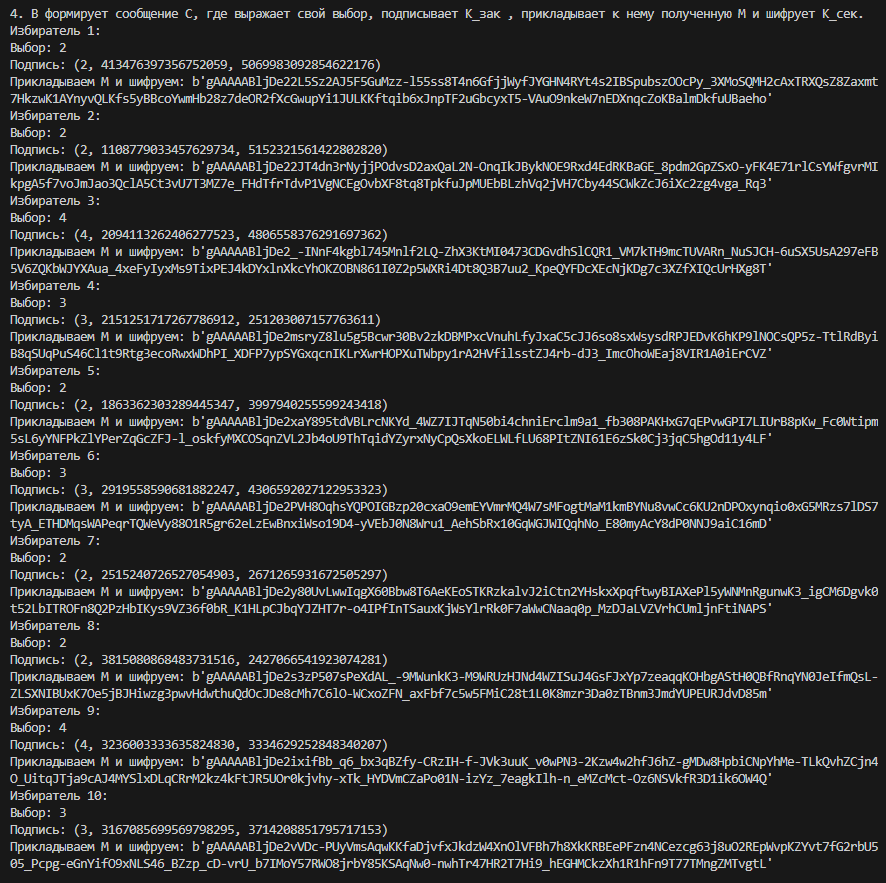


Рисунок 2 – Шаг 4



Рисунок 3 – Шаг 5

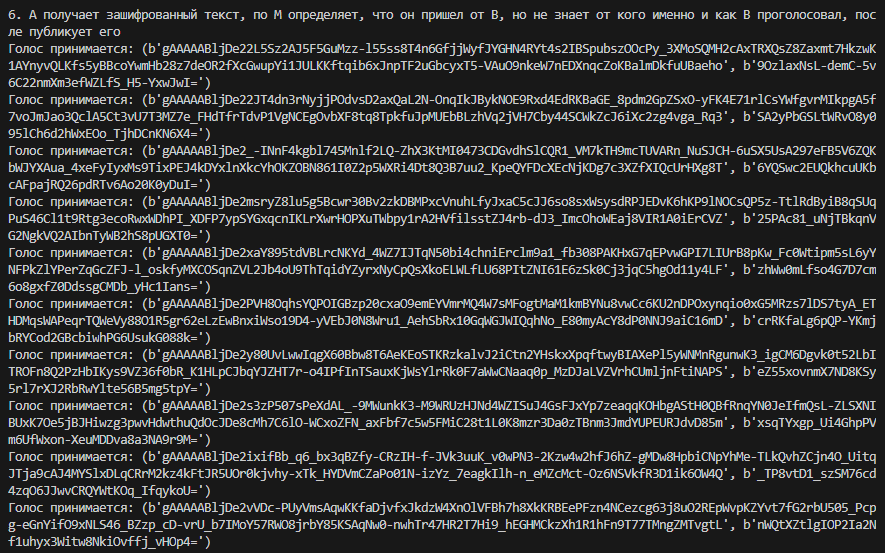


Рисунок 4 – Шаг 6



Рисунок 5 – Шаг 7

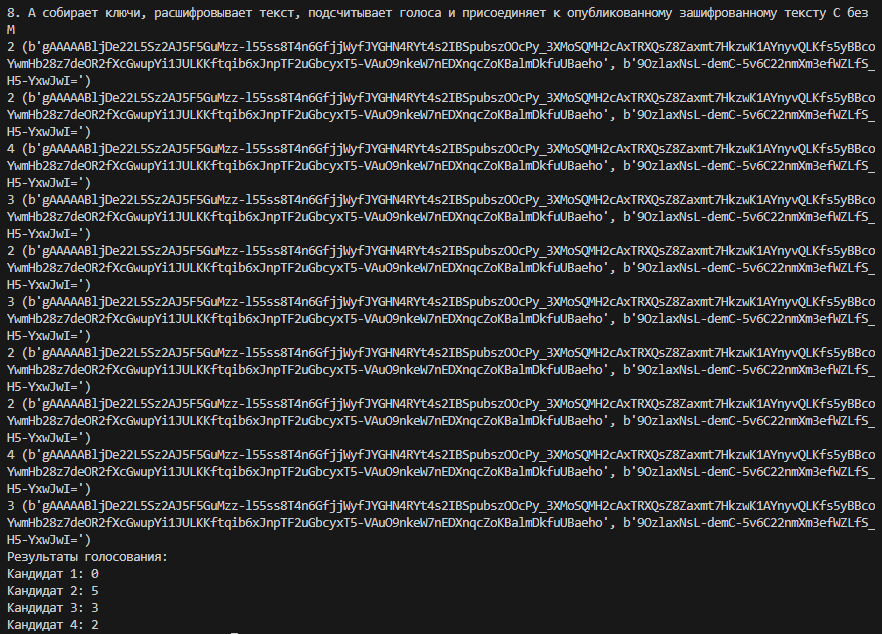


Рисунок 5 – Шаг 8

**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

from sympy import isprime

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 gen\_p(q):

res = 2 \* q

while not isprime(res + 1):

res \*= 2

return res + 1

def gen\_q(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):

res = 1

h = 2

while res == 1:

res = pow(h, (p - 1) // q, p)

h = random. randint(2, p - 2)

return res

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 check\_sign(keys, r, s):

x, y, h, q, p, g = keys

u = gcd\_ex(s, q)[1]

a = (h \* u) % q

b = (r \* u) % q

v = (pow(g, a, p) \* pow(y, b, p)) % p % q

if v == r:

return True

else:

return False

def gen\_keys(N):

C = random.choice(N)

h = abs(int(hash(str(C))))

q = gen\_q(len(bin(h)) - 2)

p = gen\_p(q)

g = gen\_g(q, p)

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

y = pow(g, x, p)

return x, y, h, q, p, g

class B:

def \_\_init\_\_(self, M, i, N, keys):

x, y, h, q, p, g = keys

C = random.choice(N)

self.C = C

self.h = h

self.q = q

self.g = g

self.p = p

self.k\_z = x

self.k\_o = y

self.k\_s = Fernet.generate\_key()

self.M = M

self.name = f"Избиратель {i}"

self.s = []

def B1(self, N):

C = self.C

#print(C)

s = 0

k = 0

r = 0

while s == 0 or r == 0:

k = random.randint(1, self.q - 1)

r = pow(self.g, k, self.p) % self.q

s = (gcd\_ex(k, self.q)[1] \* (self.h + self.k\_z\*r)) % self.q

s1 = (C, r, s)

s2 = encr(self.k\_s, (s1[0], s1[1], s1[2], self.M))

s3 = (s2, self.M)

self.s = s3

print(f"{self.name}:\nВыбор: {C}\nПодпись: {s1}\nПрикладываем М и шифруем: {s2}")

return s3

def V1(k):

m = []

for i in range (k):

m.append(Fernet.generate\_key())

return m

def A1(s, m):

if s[1] in m:

return True

else:

return False

while True:

k = int(input('Введите количество избирателей: '))

n = list(range(1, 1 + int(input('Введите количество кандидатов: '))))

#k = 10

#n = [1, 2, 3, 4, 5]

m = V1(k)

print('\n1. V отправляет М всем В до голосования')

for i, el in enumerate(m):

print(f"Избиратель {i}: {el}")

print('\n2. V отправляет А весь набор М, но без информации о том, кому они принадлежат.')

print("\n3. В создает свои ключи К\_зак , К\_отк и выкладывает в общий доступ К\_отк , а также создает секретный ключ (К\_сек ), который нужен, чтобы никто не узнал содержимое бюллетеня до нужного момента.")

Bs = []

keys = gen\_keys(n)

print("К\_зак и К\_отк будут одинаковыми для всех пользователей")

print(keys[0], keys[1])

print('\nК\_сек:')

for i in range(len(m)):

b = B(m[i], i+1, n, keys)

print(f"Избиратель {i+1}:", b.k\_s)

Bs.append(b)

print('\n4. В формирует сообщение С, где выражает свой выбор, подписывает К\_зак , прикладывает к нему полученную М и шифрует К\_сек.')

choices = []

for b in Bs:

choices.append(b.B1(n))

print('\n5. К зашифрованному тексту В прикладывает М и отправляет А.')

for b in Bs:

print(f"Избиратель {i+1}:", b.s)

print('\n6. А получает зашифрованный текст, по М определяет, что он пришел от В, но не знает от кого именно и как В проголосовал, после публикует его')

valid\_ch = []

for ch in choices:

if A1(ch, m):

print(f"Голос принимается: {ch}")

valid\_ch.append(ch)

print('\n7. Опубликованный зашифрованный текст служит информацией, чтобы В отправил К\_сек.')

keys\_mass = []

for b in Bs:

if b.s in valid\_ch:

print(f"Для сообщения {b.s} ключ - {b.k\_s}")

keys\_mass.append((b.k\_s, b.s))

print('\n8. А собирает ключи, расшифровывает текст, подсчитывает голоса и присоединяет к опубликованному зашифрованному тексту С без М')

res = []

for el in keys\_mass:

res.append(decr(el[0], el[1][0]))

for el in res:

print(el)

if check\_sign(keys, int(el[1][1]), int(el[1][2])):

res.remove(el)

res\_vote = [0] \* len(n)

for el in res:

res\_vote[int(el[0]) - 1] += 1

for i in range(len(valid\_ch)):

print(res[i][0], valid\_ch[0])

print('Результаты голосования:')

for i in range(len(res\_vote)):

print(f'Кандидат {i+1}:', res\_vote[i])