Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «КМЗИ»

Тема: “ Криптографические протоколы тайного голосования”

Выполнил:

Студент 3 курса

Группы ИИ-21

Пучинский А.А.

Проверила:

Хацкевич А.С.

Брест 2023

**Цель работы:** реализовать криптографические протоколы тайного голосования

**Варианты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер** | **Протокол** | **Алгоритм шифрования** | **Алгоритм шифрования** | **Хэш** |



**Код программы:**

**1 файл**

import Voter

import random

import string

from Crypto.PublicKey import RSA

from Crypto.Signature.pkcs1\_15 import PKCS115\_SigScheme

#просто обьявление всех глобальных переменных

voter\_list, voter\_with\_id\_blind\_and\_sign\_from\_cik = [], {}

private\_key, public\_key = None, None

#просто обьявление всех глобальных переменных

#создание случайного списка "избирателей", данные просто пустышки

def generate\_voter\_list(num\_voters):

global voter\_list

for \_ in range(num\_voters):

voter\_id = ''.join(random.choices(string.ascii\_uppercase + string.digits, k=8))

voter\_name = ''.join(random.choices(string.ascii\_uppercase, k=5))

voter\_age = random.randint(18, 90)

voter\_address = f"Street {random.randint(1, 100)}, City"

voter = {

'id': voter\_id,

'name': voter\_name,

'age': voter\_age,

'address': voter\_address

}

voter\_list.append(voter)

#генерирование пары открытый/закрытый ключ для RSA

def generate\_key\_pair():

global private\_key, public\_key

private\_key = RSA.generate(1024)

public\_key = private\_key.publickey() #создание публичного ключа на основе закрытого

# print("Private key: ", private\_key.export\_key().decode())

# print("Public key: ", public\_key.export\_key().decode())

#проверяем подпись от избирателей

def check\_signature():

global voter\_with\_id\_blind\_and\_sign\_from\_cik

for voter in Voter.voter\_with\_id\_blind\_and\_sign:

try:

PKCS115\_SigScheme(voter['public\_key']).verify(voter['blind\_m'], voter['sign\_m'])

print(f"Подпись избирателя с ID: {voter['id']} верна.")

signatured\_message\_from\_cik = PKCS115\_SigScheme(private\_key).sign(voter['blind\_m'])

voter['sign\_m\_from\_cik'] = signatured\_message\_from\_cik

except:

print(f"Подпись избирателя с ID: {voter['id']} не верна.")

voter\_with\_id\_blind\_and\_sign\_from\_cik = [{'id': voter['id'], 'blind\_m': voter['blind\_m'], 'public\_key': voter['public\_key'], 'sign\_m\_from\_cik': voter['sign\_m\_from\_cik'], 'r': voter['r'], 'M': voter['M']} for voter in Voter.voter\_with\_id\_blind\_and\_sign]

def verify\_and\_results():

candidateA, candidateB, candidateC = 0,0,0

for voter in Voter.anonim\_list:

print(f'Метка М: {voter["M"]}, Голос отдан за {voter["vote"]}')

match voter["vote"]:

case "Candidate A":

candidateA += 1

case "Candidate B":

candidateB += 1

case "Candidate C":

candidateC += 1

winner = max({"Candidate A": candidateA, "Candidate B": candidateB, "Candidate C": candidateC}.items(), key=lambda x: x[1])

print(f'Результаты: за Кандидата А отдано {candidateA} голосов,за Кандидата Б отдано {candidateB} голосов,за Кандидата С отдано {candidateC} голосов. Победил {winner[0]} с {winner[1]} голосами')

**2 файл**import CIK, Voter

def get\_cik\_key():

public\_key\_from\_CIK = CIK.public\_key #получение ключа от ЦИК

def get\_voters\_info():

global voters\_info\_in\_CSK

for voter in Voter.voters\_info\_list:

voters\_info\_in\_CSK = [voter['id'], voter['public\_key']] #получение данных(id и ключ) о избирателях

**3 файл**import CIK, Voter, CSK

print("ЦИК создает и публикует список всех избирателей...")

CIK.generate\_voter\_list(20)

print("ЦИК создает пару ключей...")

CIK.generate\_key\_pair()

print("ЦИК публикует открытый ключ в ЦСК...")

CSK.get\_cik\_key()

print("Избиратели публикуют свои ID и ключ в ЦСК...")

Voter.generate\_specific\_voters\_data\_list()

CSK.get\_voters\_info()

print("Избиратели генерируют свою уникальную метку...")

print("Избиратели ослепляют и подписывают свои метки и отправляют в ЦИК...")

Voter.blinding\_protocol()

print("ЦИК проверяет ЭЦП избирателей и отправляет избирателям слепую ЭЦП...")

CIK.check\_signature()

print("Избиратели снимают закрывающий множитель с ЭЦП комиссии...")

Voter.remove\_r()

print("Избиратели голосуют...")

print("Отправка анонимных результатов в ЦИК...")

print("ЦИК проверяет данные и публикует метку и бюллитень и выводит результат")

CIK.verify\_and\_results()

**4 файл**

import CIK

import random

from Crypto.PublicKey import RSA

from Crypto.Signature.pkcs1\_15 import PKCS115\_SigScheme

from Crypto.Hash import SHA1

#просто обьявление всех глобальных переменных

voters\_info\_list, voter\_with\_id\_blind\_and\_sign = {}, {}

#просто обьявление всех глобальных переменных

#бюллитень с кандидатами

bulliten = ["Candidate A", "Candidate B", "Candidate C"]

generated\_marks = set() #для отследивания меток M(мало ли одинковые сгенерятся)

#генерирование меток M

def generate\_unique\_mark():

generated = False

while generated == False:

random\_mark = random.randint(10\*\*9, 10\*\*10 - 1)

if random\_mark not in generated\_marks:

generated\_marks.add(random\_mark)

generated = True

return random\_mark

generated\_multi = set() #то же что и сет выше, но для множителя

#генерирование уникальных закрытых множителей

def generate\_unique\_closed\_miltiplier():

generated = False

while generated == False:

random\_multi = random.randint(10\*\*4, 10\*\*5 - 1)

if random\_multi not in generated\_multi:

generated\_multi.add(random\_multi)

generated = True

return random\_multi

#генерация открытого\закрытого ключа и метки M для каждого избирателя

def generate\_specific\_voters\_data\_list():

global voters\_info\_list

voters\_list\_from\_CIK = CIK.voter\_list

i = 1

for voter in voters\_list\_from\_CIK:

private\_key = RSA.generate(1024)

public\_key = private\_key.publickey()

M = generate\_unique\_mark()

r = generate\_unique\_closed\_miltiplier()

voter['private\_key'] = private\_key

voter['public\_key'] = public\_key

voter['M'] = M

voter['r'] = r

print(f'Избиратель {i} сгенерирован')

i += 1

#создание списка с такими данными для дальнейшей работы

voters\_info\_list = [{'id': voter['id'], 'public\_key': voter['public\_key'], 'private\_key': voter['private\_key'], 'M': voter['M'], 'r': voter['r']} for voter in voters\_list\_from\_CIK]

#протокол для "ослепления" метки M

def blinding\_protocol():

global voter\_with\_id\_blind\_and\_sign

#используем уникальный закрывающий множитель r, открытый ключ из ЦИК для скрытия метки M

for voter in voters\_info\_list:

blinded\_message = (voter['r'] \*\* CIK.public\_key.e \* voter['M']) % CIK.public\_key.n

#перевод в байты для хэширования

blinded\_message\_bytes = blinded\_message.to\_bytes((blinded\_message.bit\_length() + 7) // 8, 'big')

#хэшируем полученное сообщение

hashed\_blinded\_message = SHA1.new(blinded\_message\_bytes)

#подписываем захэшированное сообщение с закрытым ключом каждого избирателя

signatured\_message = PKCS115\_SigScheme(voter['private\_key']).sign(hashed\_blinded\_message)

voter['blind\_m'] = hashed\_blinded\_message

voter['sign\_m'] = signatured\_message

#создаем список с полученными данными

voter\_with\_id\_blind\_and\_sign = [{'id': voter['id'], 'blind\_m': voter['blind\_m'], 'public\_key': voter['public\_key'], 'sign\_m': voter['sign\_m'], 'private\_key': voter['private\_key'], 'r': voter['r'], 'M': voter['M']} for voter in voters\_info\_list]

def remove\_r():

global anonim\_list

for voter in CIK.voter\_with\_id\_blind\_and\_sign\_from\_cik:

hashed\_message\_from\_cik = SHA1.new()

hashed\_message\_from\_cik.update(voter['sign\_m\_from\_cik'])

#вычисление обратного значения r

r\_inv = pow(voter['r'], -1, CIK.public\_key.n)

#ЭЦП комисии после снятия закрытого множителя r

DS\_int = int.from\_bytes(voter['sign\_m\_from\_cik'], byteorder='big')

DS = (DS\_int \* r\_inv) % CIK.public\_key.n

voter['DS'] = DS

vote = random.choice(bulliten)

voter['vote'] = vote

#список с данными для отправки

anonim\_list = [{'M': voter['M'], 'vote': voter['vote'], 'DS': voter['DS']} for voter in CIK.voter\_with\_id\_blind\_and\_sign\_from\_cik]

**Результат работы**

ЦИК создает и публикует список всех избирателей...  
ЦИК создает пару ключей...  
ЦИК публикует открытый ключ в ЦСК...  
Избиратели публикуют свои ID и ключ в ЦСК...  
Избиратель 1 сгенерирован  
Избиратель 2 сгенерирован  
Избиратель 3 сгенерирован  
Избиратель 4 сгенерирован  
Избиратель 5 сгенерирован  
Избиратель 6 сгенерирован  
Избиратель 7 сгенерирован  
Избиратель 8 сгенерирован  
Избиратель 9 сгенерирован  
Избиратель 10 сгенерирован  
Избиратель 11 сгенерирован  
Избиратель 12 сгенерирован  
Избиратель 13 сгенерирован  
Избиратель 14 сгенерирован  
Избиратель 15 сгенерирован  
Избиратель 16 сгенерирован  
Избиратель 17 сгенерирован  
Избиратель 18 сгенерирован  
Избиратель 19 сгенерирован  
Избиратель 20 сгенерирован  
Избиратели генерируют свою уникальную метку...  
Избиратели ослепляют и подписывают свои метки и отправляют в ЦИК...  
ЦИК проверяет ЭЦП избирателей и отправляет избирателям слепую ЭЦП...  
Подпись избирателя с ID: 1SRQWW04 верна.  
Подпись избирателя с ID: YW9WS9WC верна.  
Подпись избирателя с ID: WPD8KXC3 верна.  
Подпись избирателя с ID: LCNHKIXS верна.  
Подпись избирателя с ID: 0RA87AO9 верна.  
Подпись избирателя с ID: FLBO5X59 верна.  
Подпись избирателя с ID: OQ662TV2 верна.  
Подпись избирателя с ID: 5V8SBCX3 верна.  
Подпись избирателя с ID: 5HN9O91P верна.  
Подпись избирателя с ID: ZKP15WBL верна.  
Подпись избирателя с ID: NPP8E61V верна.  
Подпись избирателя с ID: 6SIE5C3D верна.  
Подпись избирателя с ID: J43ZBXQ6 верна.  
Подпись избирателя с ID: KV83F50C верна.  
Подпись избирателя с ID: Z9FOFC2S верна.  
Подпись избирателя с ID: 5WDYK880 верна.  
Подпись избирателя с ID: UAQLGI45 верна.  
Подпись избирателя с ID: CTF9XI5C верна.  
Подпись избирателя с ID: YPIJFJLR верна.  
Подпись избирателя с ID: Y1NEZRI4 верна.  
Избиратели снимают закрывающий множитель с ЭЦП комиссии...  
Избиратели голосуют...  
Отправка анонимных результатов в ЦИК...  
ЦИК проверяет данные и публикует метку и бюллитень и выводит результат  
Метка М: 3061007942, Голос отдан за Candidate A  
Метка М: 3847953619, Голос отдан за Candidate C  
Метка М: 3390435953, Голос отдан за Candidate C  
Метка М: 2290427077, Голос отдан за Candidate C  
Метка М: 3924874226, Голос отдан за Candidate C  
Метка М: 8829798110, Голос отдан за Candidate B  
Метка М: 6059683324, Голос отдан за Candidate C  
Метка М: 1887303987, Голос отдан за Candidate B  
Метка М: 7640054527, Голос отдан за Candidate A  
Метка М: 3863104989, Голос отдан за Candidate C  
Метка М: 3763626872, Голос отдан за Candidate B  
Метка М: 4288080997, Голос отдан за Candidate A  
Метка М: 1758372467, Голос отдан за Candidate B  
Метка М: 8738212411, Голос отдан за Candidate A  
Метка М: 1561528293, Голос отдан за Candidate C  
Метка М: 7399587919, Голос отдан за Candidate B  
Метка М: 4961412754, Голос отдан за Candidate A  
Метка М: 3568162467, Голос отдан за Candidate A  
Метка М: 7930518055, Голос отдан за Candidate B  
Метка М: 5399629967, Голос отдан за Candidate C  
Результаты: за Кандидата А отдано 6 голосов,за Кандидата Б отдано 6 голосов,за Кандидата С отдано 8 голосов. Победил Candidate C с 8 голосами

**Вывод:** реализовал криптографический протокол тайного голосования (Протоколы двух агентств Фудзиока-Окамото-Охта и Sensus)