Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Криптографические методы защиты информации»

Тема: «Криптографические протоколы тайного голосования»

Выполнила:

Студент 3 курса

Группы ИИ-23

Макаревич Н.Р.

Проверила:

Хацкевич А. С.

Брест 2024

**Цель:** создать программу, которая реализует работу электронного голосования

**Ход работы:**

**Вариант 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Протоколы двух агентств Фудзиока-Окамото-Охта и Sensus | RC-2 | Вижинера | SHA-1 |

**Код работы класса для работы с электронным голосованием:**

import hashlib

import os

class Registrar:

def \_\_init\_\_(self):

self.public\_key = self.generate\_key()

self.private\_key = self.generate\_key()

def generate\_key(self):

return os.urandom(16)

def sign(self, message):

return hashlib.sha1(message.encode()).hexdigest()

def verify(self, message, signature):

return signature == self.sign(message)

class Voter:

def \_\_init\_\_(self, voter\_id):

self.voter\_id = voter\_id

self.public\_key = self.generate\_key()

self.private\_key = self.generate\_key()

self.secret\_key = os.urandom(16)

def generate\_key(self):

return os.urandom(16)

def vigenere\_encrypt(self, message, key):

encrypted = []

key\_length = len(key)

for i, char in enumerate(message):

if char.isalpha():

shift = ord(key[i % key\_length].lower()) - ord('a')

if char.islower():

encrypted\_char = chr((ord(char) - ord('a') + shift) % 26 + ord('a'))

else:

encrypted\_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))

encrypted.append(encrypted\_char)

else:

encrypted.append(char)

return ''.join(encrypted)

def blind(self, encrypted\_ballot, closing\_factor):

return f"{closing\_factor}:{encrypted\_ballot}"

def sign(self, message):

return hashlib.sha1(message.encode()).hexdigest()

class Counter:

def \_\_init\_\_(self, registrar):

self.registrar = registrar

def verify\_signature(self, signature, message):

return self.registrar.verify(message, signature)

def vigenere\_decrypt(self, encrypted\_message, key):

decrypted = []

key\_length = len(key)

for i, char in enumerate(encrypted\_message):

if char.isalpha():

shift = ord(key[i % key\_length].lower()) - ord('a')

if char.islower():

decrypted\_char = chr((ord(char) - ord('a') - shift) % 26 + ord('a'))

else:

decrypted\_char = chr((ord(char) - ord('A') - shift) % 26 + ord('A'))

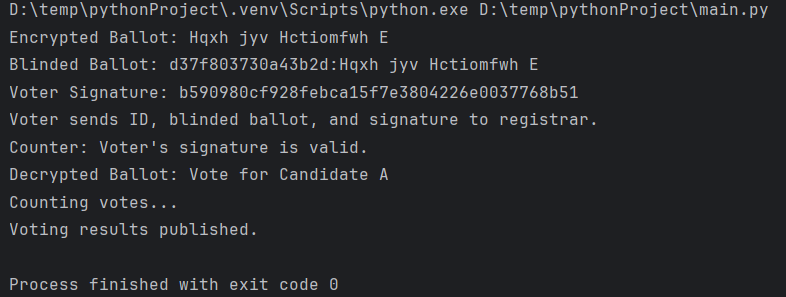
decrypted.append(decrypted\_char)

else:

decrypted.append(char)

return ''.join(decrypted)

**Результат работы программы:**



**Вывод:** создал программу, которая реализует работу с эллиптическими кривыми