Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# ОТЧЕТ

# О ЛАБОРАТОРНОЙ РАБОТЕ

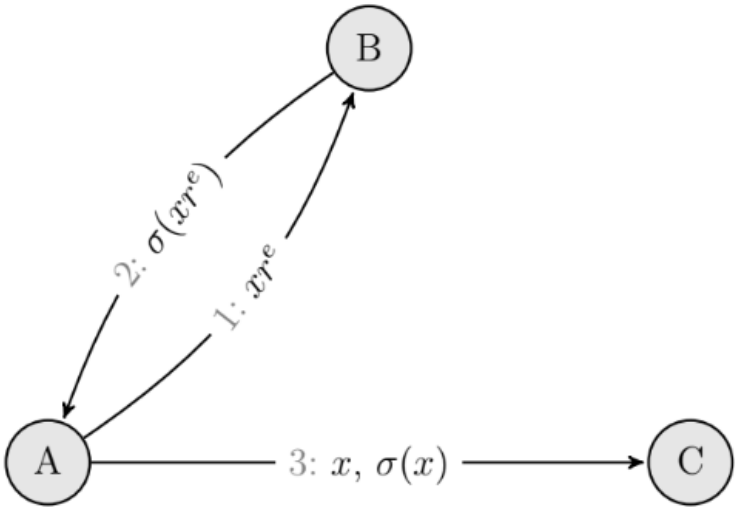
Лабораторная работа №4
по теме:
«Создание и верификация цифровой подписи»

Исполнитель, студент группы 201-361
_____ Н.А. Фельдбуш

Москва, 2023

**Постановка задачи**:

Реализуйте простое клиент-серверное приложение, позволяющее аккумулировать короткие анонимные сообщения (систему электронного голосования) согласно следующей схеме:




Здесь: A – пользователь (избиратель), B – регистратор, C – счетчик, x – сообщение (голос), r – известное только участнику A случайное число, (e, n) – открытый ключ банка. Пренебрегите реализацией правильных механизмов распределения, хранения и сертификации ключей.

**Ход работы:**

Выполним поставленную задачу при помощи средств языка программирования Python. Перед началом следует обратить внимание на порядок операций, выполняемых пользователем, регистратором и счетчиком. В первую очередь регистратор формирует у себя закрытый ключ, по которому формирует открытый который позже передает пользователю. Затем пользователь с помощью модуля открытого ключа формирует случайное число r. После этого используя модуль экспоненту и число r ослепляет сообщение и отправляет его регистратору, который в свою очередь подписывает его с помощью модуля и экспоненты закрытого ключа, а затем

отправляет обратно пользователю. Пользователь проводит операцию обратную операции ослепления и получает подписанный закрытым ключом исходный файл, который затем в комбинации с изначальным файлом отправляется счетчику. Счетчик с помощью функции подтверждения подписи проверяет, действительна ли она или нет.

Итак:

1. Регистратор формирует закрытый и открытый ключ (рис. 1-2):


Рисунок 1 – Закрытый ключ

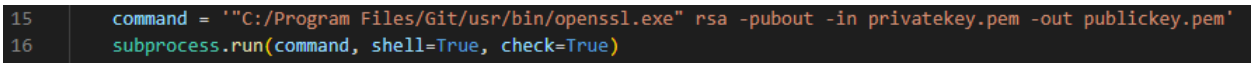



Рисунок 2 – Открытый ключ

2. Передача открытого ключа пользователю (рис. 3):

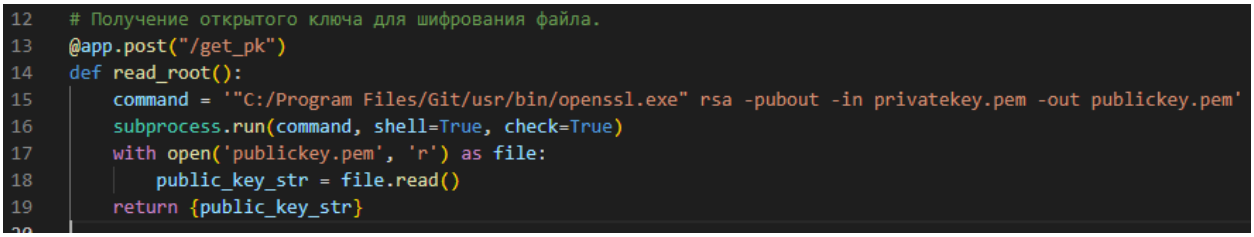



Рисунок 3 – Передача ключа

3. Получение числа r (рис. 4):

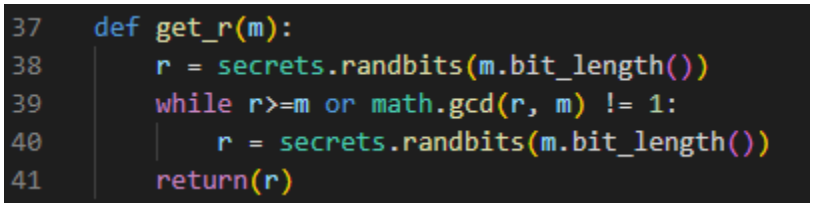



Рисунок 4 – Число r

4. Ослепление сообщения (рис. 5):

```
43    def blind_message(message, r, e, n):
44        return (message * pow(r, e, n)) % n
```

Рисунок 5 – Функция ослепления

5. Отправка сообщения (рис. 6):

```
25    message_to_send = blind_message(message,r,exponent,modulus)
26    req = requests.post("http://localhost:5000/get_vote", json = [message_to_send])
```

Рисунок 6 – Отправка сообщения

6. Подписание с помощью модуля и экспоненты закрытого ключа (рис. 7):

```
22  @app.post("/get_vote")
23  def authorize(request: Request, message: List[int]):
24      p_exponent = get_private_exponent()
25      modulus, exponent = get_modulus_and_exponent()
26      return(sign_blinded_message(message[0], p_exponent, modulus))
27
28
29  def sign_blinded_message(message, privare_exponent, modulus):
30      return pow(message, privare_exponent, modulus)
31
```

Рисунок 7 – Подписание с помощью модуля и экспоненты

7. Получение подписанного сообщения клиентом и операция обратная ослеплению (рис. 8):

```
26    req = requests.post("http://localhost:5000/get_vote", json = [message_to_send])
27
28    message_unblind= unblind_signature(int(req.json()),r,modulus)
29
30    message_results=[]
31    message_results.append(message)
32    message_results.append(message_unblind)
```

Рисунок 8 – Операция обратная ослеплению

8. Отправка файлов счетчику (рис. 9):

```
31    message_results.append(message)
32    message_results.append(message_unblind)
33
34    req = requests.post("http://localhost:5001/count", json = message_results)
```

Рисунок 9 – Отправка изначального файла и подписанного

9. Функция подтверждения у счетчика (рис. 10):

```
27    def verify_signature(signed_m, m, e, n):
28        return pow(signed_m, e, n) == m
```

Рисунок 10 – Функция подтверждения

Проверим работу приложения. Зададим количество избирателей равным трем участникам, запустим все три класса в необходимом порядке и рассмотрим содержимое консоли (рис.14). Консоль разделена на три части: слева – ClientA, по середине – ServerB, справа – ServerC. Можем заметить, как клиент отправляет замаскированное сообщение серверу B, сервер B его обрабатывает, а затем сервер C получает голос (рис. 11).

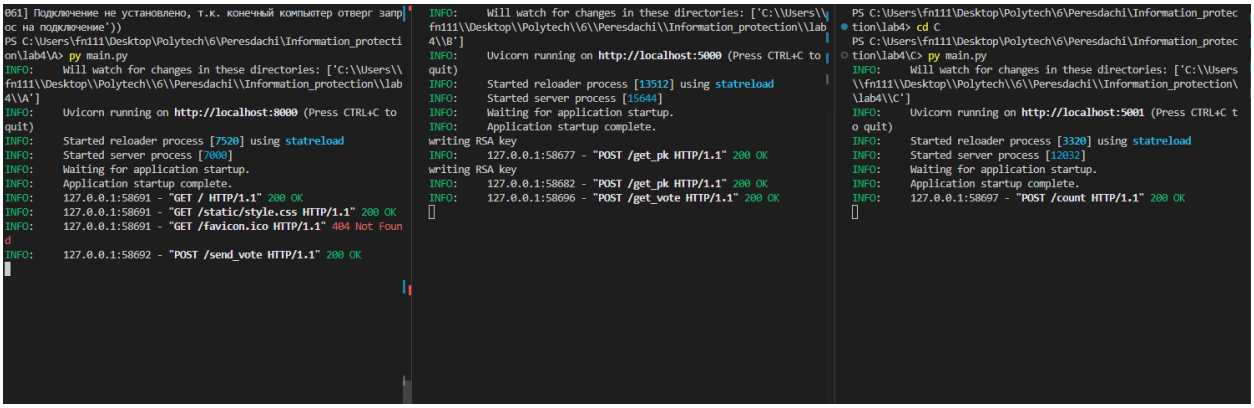



Рисунок 11 – Работа приложения

Также дополнительно визуализирован подсчет голосов (рис. 12):

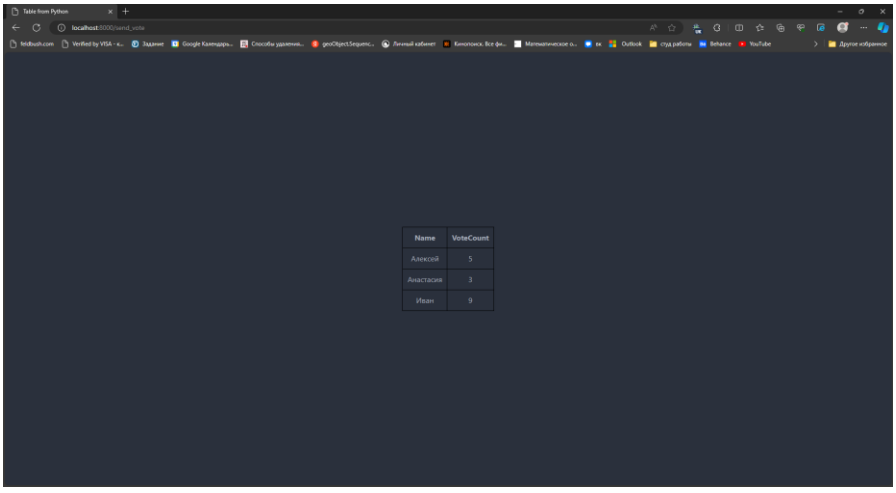



Рисунок 12 – Работа приложения