

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ
ПО ДИСЦИПЛИНЕ
«Защита информации»

Руководители

_____ Е.А. Харченко

Москва 2023

СПИСОК ИСПОЛНИТЕЛЕЙ

Студ. группы 201-361

_____ А.Е. Сильченко

Принцип работы программы:

Клиент выбирает за кого он хочет проголосовать, в зависимости от выбора формируется текстовый файл, с кандидатом за которого он проголосовал. Далее на сервере А генерируются публичный и приватный ключ, публичный ключ отправляется на клиент. Клиент принимает публичный ключ и шифрует сообщение, затем это зашифрованное сообщение передается на сервер А. На сервере А сообщение расшифровывается и подписывается, подписанное сообщение отправляется на клиент. Далее клиент отправляет открытый ключ, подписанное сообщение и сообщение на сервер Б, сервер Б проверяет подпись, если подпись прошла проверку, то голос добавляется одному из кандидатов, и результаты(в данном случае 3 переменных), отправляются на клиент для отображения текущих результатов голосования.

Ниже приведены скриншоты классов.

Client.java

```

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
    Integer first = 0;
    Integer second = 0;
    Integer third = 0;
    String message = "";
    boolean stop = true;
    while (stop) {
        //подключение к серверам
        Socket socket_a = new Socket( host: "localhost", port: 12345);
        Socket socket_b = new Socket( host: "localhost", port: 1234);

        //процесс голосования
        System.out.println("Выберите кандидата, за которого вы хотите проголосовать: ");
        System.out.println("1 - Кандидат 1");
        System.out.println("2 - Кандидат 2");
        System.out.println("3 - Кандидат 3");
        Scanner scanner = new Scanner(System.in);
        String vibor = scanner.nextLine();
        if (vibor.equals("1")) {
            message = "Кандидат 1";
            FileWriter writer = new FileWriter( fileName: "message.txt");
            writer.write(message);
            writer.close();
        } else if (vibor.equals("2")) {
            message = "Кандидат 2";
            FileWriter writer = new FileWriter( fileName: "message.txt");
            writer.write(message);
            writer.close();
        } else if (vibor.equals("3")) {
            message = "Кандидат 3";
            FileWriter writer = new FileWriter( fileName: "message.txt");
            writer.write(message);
            writer.close();
        }

        //получаем открытый ключ и записываем его в файл
        ObjectInputStream in_a_1 = new ObjectInputStream(socket_a.getInputStream());
        byte[] open_key = (byte[]) in_a_1.readObject();
        try {
            FileOutputStream fos = new FileOutputStream( name: "open.pem");
            fos.write(open_key);
        }
    }
}

```

```

//получаем открытый ключ и записываем его в файл
ObjectInputStream in_a_1 = new ObjectInputStream(socket_a.getInputStream());
byte[] open_key = (byte[]) in_a_1.readObject();
try {
    FileOutputStream fos = new FileOutputStream( name: "open.pem");
    fos.write(open_key);
    fos.close();
} catch (IOException e) {
    System.out.println("Error writing file: " + e);
}

// Шифруем файл сообщения
Process encryptProcess = Runtime.getRuntime().exec( command: "openssl rsautl -encrypt -inkey open.pem -pubin -in message.txt -out message.enc");
encryptProcess.waitFor();

//отправка зашифрованного сообщения на сервер
DataOutputStream out_a = new DataOutputStream(socket_a.getOutputStream());
ObjectOutputStream objectOut_a = new ObjectOutputStream(out_a);
byte[] message_1 = Files.readAllBytes(Paths.get( first: "message.enc"));
objectOut_a.writeObject(message_1);

ObjectInputStream in_a = new ObjectInputStream(socket_a.getInputStream());
byte[] sign = (byte[]) in_a.readObject(); //подписанное сообщение

DataOutputStream out_b = new DataOutputStream(socket_b.getOutputStream());
ObjectOutputStream objectOut = new ObjectOutputStream(out_b);
byte[] message_2 = Files.readAllBytes(Paths.get( first: "message.txt"));
objectOut.writeObject(message_2); //отправка сообщения на сервер б
objectOut.writeObject(sign); //отправка подписанного сообщения на сервер б
objectOut.writeObject(open_key); //отправка открытого ключа на сервер б

```

```

//получение данных от сервера(счетчик голосов)
DataInputStream in_b = new DataInputStream(socket_b.getInputStream());
first = in_b.readInt();
second = in_b.readInt();
third = in_b.readInt();
System.out.println("Кол-во голосов на данный момент: ");
System.out.println("За первого кандидата: " + first);
System.out.println("За второго кандидата: " + second);
System.out.println("За третьего кандидата: " + third);

System.out.println("Продолжить голосование? 1-да, 0-нет");
scanner = new Scanner(System.in);
vibor = scanner.nextLine();
if (vibor.equals("0")) {
    stop = false;
}

socket_a.close();
socket_b.close();
}

System.out.println("Голосование завершено! Результаты:");
System.out.println("За первого кандидата: " + first);
System.out.println("За второго кандидата: " + second);
System.out.println("За третьего кандидата: " + third);
// Удаляем временные файлы
Files.deleteIfExists(Paths.get( first: "message.txt"));
Files.deleteIfExists(Paths.get( first: "open.pem"));
Files.deleteIfExists(Paths.get( first: "message.enc"));

```

ServerA.java

```
public class ServerA {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        while (true) {
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Waiting for client...");
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected");

            // Создаем открытый и закрытый ключ
            Process genpkeyProcess = Runtime.getRuntime().exec("openssl genpkey -algorithm RSA -out privatekey.pem -pkeyopt rsa_keygen_bits:1024");
            genpkeyProcess.waitFor();
            Process rsaProcess = Runtime.getRuntime().exec("openssl rsa -pubout -in privatekey.pem -out publickey.pem");
            rsaProcess.waitFor();

            DataOutputStream out_1 = new DataOutputStream(clientSocket.getOutputStream());
            ObjectOutputStream objectOut_1 = new ObjectOutputStream(out_1);
            //отправить ключ клиенту
            byte[] open_key = Files.readAllBytes(Paths.get("publickey.pem"));
            objectOut_1.writeObject(open_key); //открытый ключ

            //получение сообщения и сохранение в файл
            ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());
            byte[] message = (byte[]) in.readObject();

            try {
                FileOutputStream fos = new FileOutputStream("m_ec.txt");
                fos.write(message);
                fos.close();
            } catch (IOException e) {
                System.out.println("Error writing file: " + e);
            }

            //расшифровка сообщения
            Process decryptProcess = Runtime.getRuntime().exec("openssl rsautl -decrypt -inkey privatekey.pem -in message.enc -out message.dec");
            decryptProcess.waitFor();

            DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());
            ObjectOutputStream objectOut = new ObjectOutputStream(out);

            //расшифровка сообщения
            Process decryptProcess = Runtime.getRuntime().exec("openssl rsautl -decrypt -inkey privatekey.pem -in message.enc -out message.dec");
            decryptProcess.waitFor();

            DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());
            ObjectOutputStream objectOut = new ObjectOutputStream(out);

            // Подписываем сообщение
            Process dgstProcess = Runtime.getRuntime().exec("openssl dgst -sha256 -sign privatekey.pem -out signature.bin message.dec");
            dgstProcess.waitFor();

            byte[] sign = Files.readAllBytes(Paths.get("signature.bin"));
            objectOut.writeObject(sign); //подписанное сообщение

            //удаляем временные файлы
            Files.deleteIfExists(Paths.get("privatekey.pem"));
            Files.deleteIfExists(Paths.get("publickey.pem"));
            Files.deleteIfExists(Paths.get("message.enc"));
            Files.deleteIfExists(Paths.get("signature.bin"));
            Files.deleteIfExists(Paths.get("m_ec.txt"));
            Files.deleteIfExists(Paths.get("message.dec"));

            clientSocket.close();
            serverSocket.close();
        }
    }
}
```

ServerB.java

```
public class ServerB {  
    public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException {  
  
        //Integer v = 0;  
        Integer first = 0;  
        Integer second = 0;  
        Integer third = 0;  
        while (true) {  
  
            ServerSocket serverSocket = new ServerSocket( port: 1234);  
            System.out.println("Waiting for client...");  
            Socket clientSocket = serverSocket.accept();  
            System.out.println("Client connected");  
  
            ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());  
            byte[] message_1 = (byte[]) in.readObject();  
            byte[] signature = (byte[]) in.readObject();  
            byte[] openkey = (byte[]) in.readObject();  
  
            //получение выбора  
            // DataInputStream in_1 = new DataInputStream(clientSocket.getInputStream());  
            // v = in_1.readInt();  
  
            //записываем полученные данные в файлы  
            try {  
                FileOutputStream fos = new FileOutputStream( name: "m.txt");  
                fos.write(message_1);  
                fos.close();  
            } catch (IOException e) {  
                System.out.println("Error writing file: " + e);  
            }  
  
            try {  
                FileOutputStream fos = new FileOutputStream( name: "s.bin");  
                fos.write(signature);  
                fos.close();  
            } catch (IOException e) {  
                System.out.println("Error writing file: " + e);  
            }  
        }  
    }  
}
```

```

try {
    FileOutputStream fos = new FileOutputStream( name: "k.pem");
    fos.write(openkey);
    fos.close();
} catch (IOException e) {
    System.out.println("Error writing file: " + e);
}

//получить содержимое сообщения
String filePath = "m.txt";
String fileContents = new String(Files.readAllBytes(Paths.get(filePath)), StandardCharsets.UTF_8);

boolean result = verifySignature(); //проверяем подпись
if(result){
    if(fileContents.equals("Кандидат 1")){
        first++;
    }
    else if(fileContents.equals("Кандидат 2")){
        second++;
    }
    else if(fileContents.equals("Кандидат 3")){
        third++;
    }
}

//отправка данных клиенту(счетчик голосов)
DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());
out.writeInt(first);
out.writeInt(second);
out.writeInt(third);
clientSocket.close();
serverSocket.close();

//удаляем временные файлы
Files.deleteIfExists(Paths.get( first: "k.pem"));
Files.deleteIfExists(Paths.get( first: "s.bin"));
Files.deleteIfExists(Paths.get( first: "m.txt"));

```

```

1 usage
public static boolean verifySignature() throws IOException, InterruptedException {
    // Проверяем подпись
    Process verifyProcess = Runtime.getRuntime().exec( command: "openssl dgst -sha256 -verify k.pem -signature s.bin m.txt");
    verifyProcess.waitFor();
    String verifyOutput = new String(verifyProcess.getInputStream().readAllBytes());
    if (verifyOutput.contains("OK")) {
        return true;
    } else {
        return false;
    }
}

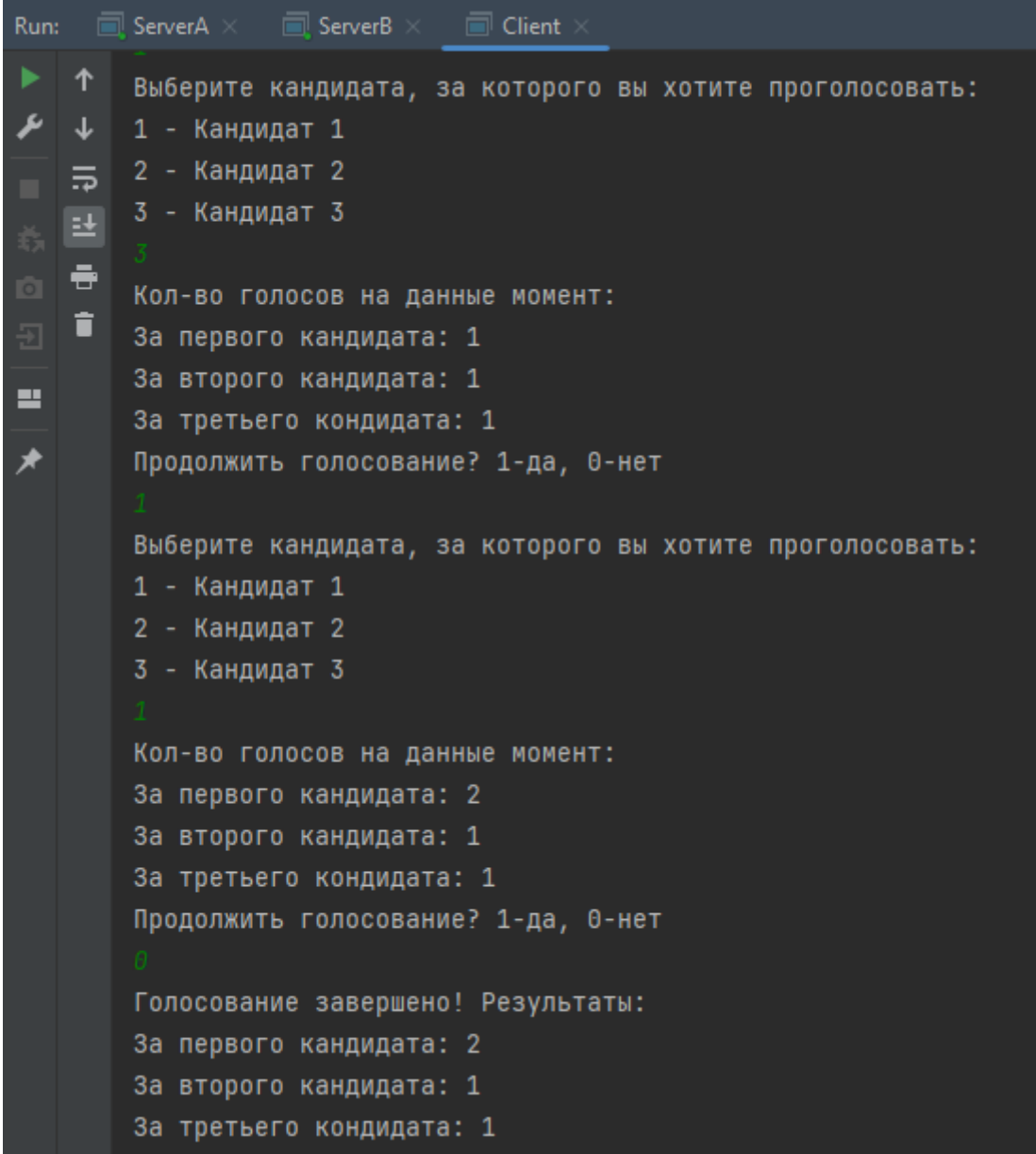
```


Итог:

Для запуска программы в начале запускаются Сервера А и Б, далее ожидают подключения клиента, далее запускается клиент, подключается к серверам и проводится голосование.

В процессе передачи данных между серверами и клиентом, а также при выполнении команд OpenSSL генерируются файлы, которые после проведения голосования удаляются.

Результат проведения голосования:



```
Run: ServerA x ServerB x Client x
Выберите кандидата, за которого вы хотите проголосовать:
1 - Кандидат 1
2 - Кандидат 2
3 - Кандидат 3
3
Кол-во голосов на данный момент:
За первого кандидата: 1
За второго кандидата: 1
За третьего кандидата: 1
Продолжить голосование? 1-да, 0-нет
1
Выберите кандидата, за которого вы хотите проголосовать:
1 - Кандидат 1
2 - Кандидат 2
3 - Кандидат 3
1
Кол-во голосов на данный момент:
За первого кандидата: 2
За второго кандидата: 1
За третьего кандидата: 1
Продолжить голосование? 1-да, 0-нет
0
Голосование завершено! Результаты:
За первого кандидата: 2
За второго кандидата: 1
За третьего кандидата: 1
```