Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-

вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Заведующий каф. КИБЭВС

\_\_\_\_\_\_\_\_\_ А.А. Шелупанов

\_\_\_\_.\_\_\_\_.\_\_\_\_\_\_\_\_ дата

ПРОГРАММНОЕ СРЕДСТВО СОЗДАНИЯ ЭЛЕКТРОННОЙ ПОДПИСИ И ЦИФРОВЫХ СЕРТИФИКАТОВ

Курсовая работа по дисциплине ««Программно-аппаратные средства обеспечения информационной безопасности»

Пояснительная записка

Студент гр. 727-1

\_\_\_\_\_\_\_\_\_ Лодонова Б.С.

Принял:

Доцент кафедры КИБЭВС

\_\_\_\_\_\_\_\_ И.А.Рахманенко

\_\_\_\_\_\_\_\_

Реферат

Пояснительная записка содержит 47 листов, 32 рисунков, 2 таблиц, 8 использованных источников, 3 приложения.

Электронная подпись, Электронный Сертификат,

Полное название системы: Программное средство создания электронной подписи и цифровых сертификатов.

Краткое наименование системы: TakingSign.

Цель работы – разработка программного средства создания электронной подписи и цифровых сертификатов.

В результате работы было создано приложение, которое корректно выполняет свои функции.

Разработка производилась на языках программирования С# и Python 3, в качестве интегрированной среды разработки использовались средства и службы для разработчиков Visual Studio 2019 и Jupyter Notebook, в качестве среды функционирования использовалась операционная система Windows 10, в качестве вспомогательных инструментов разработки использовались библиотеки искусственного интеллекта Tensorflow и Keras, библиотека работы с данными Pandas, поставщик служб шифрования Microsoft Enhanced Cryptographic Provider.

Пояснительная записка к курсовой работе выполнена в текстовом редакторе Microsoft Word 2016.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Заведующий кафедрой КИБЭВС

д-р техн. наук, профессор

\_\_\_\_\_\_\_\_\_\_\_\_А.А. Шелупанов

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

ЗАДАНИЕ

На курсовую работу по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности» студенту Лодоновой Бальжит Соктоевне группы 727 факультета Безопасности

1. Тема работы:

«Программное средство создания электронной подписи и цифровых сертификатов».

2. Срок сдачи студентом законченной работы:

8 июня 2021г.

3. Исходные данные к работе:

* операционная система Windows 10;
* библиотеки искусственного интеллекта Tensorflow и Keras;
* библиотека работы с данными Pandas;
* среды программирования Visual Studio 2019 и Jupyter Notebook;
* Языковые средства: C#, Python 3;
* Microsoft Enhanced Cryptographic Provider;

4. Содержание пояснительной записки:

* + титульный лист;
  + реферат;
  + техническое задание;
  + введение;
  + обзор предметной области;
  + проектирование архитектуры системы
  + обоснование выбора программных средств;
  + описание модулей прикладной программы;
  + описание алгоритмов работы;
  + тестирование;
  + заключение;
  + список литературы;
  + руководство пользователя.;
  + приложение.

5. Перечень подлежащих разработке вопросов:

* Разработка интерфейса взаимодействия с пользователем. Модуль предназначен для предоставления пользователю инструментов взаимодействия с функционалом программы;
* Разработка модуля съема данных, вводимых координатным устройством. Модуль необходим для съема значений, вводимых устройством, и дальнейшего приведения их к виду, необходимому для обработки нейронной сетью;
* Разработка архитектуры нейронной сети. Разработка архитектуры нейронной сети, предназначенной для обработки входных данных и создания на их основе сида для генерации ключей и сертификатов;
* Разработка модуля генерации ключей и сертификатов;
* Разработка модуля шифрования и расшифровывания по подписи посредством сгенерированных ключей и сертификатов.

6. Задание выдано:

Руководитель: к.т.н., доцент кафедры БИС

И.А. Рахманенко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

(подпись)

7. Задание принято к исполнению:

Студент группы 727

Б. С. Лодонова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

(подпись)

Содержание

Оглавление

[1 Введение 7](#_Toc73978629)

[2 Обзор предметной области 8](#_Toc73978630)

[3 Проектирование архитектуры системы 11](#_Toc73978631)

[4 Обоснование выбора программных средств 13](#_Toc73978632)

[5 Описание модулей прикладной программы 14](#_Toc73978633)

[6 Описание алгоритмов работы 23](#_Toc73978634)

[7 Тестирование 29](#_Toc73978635)

[Заключение 35](#_Toc73978636)

[Список литературы 36](#_Toc73978637)

[Приложение А 37](#_Toc73978638)

[Приложение Б 45](#_Toc73978639)

# 1 Введение

В связи с быстрым ростом онлайн-взаимодействий в Интернете, такие функции службы безопасности, как аутентификация, проверка неотказуемости, целостности и сохранение конфиденциальности, являются очень важными требованиями безопасности для безопасной транзакции. Шифрование информации - один из важнейших способов защиты информации в киберпространстве. Для достижения этих целей цифровая подпись является наиболее эффективным инструментом, использующим криптографию.

Согласно ФЗ №63 об электронной подписи [], ЭЦП это информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию. Ключ проверки этой ЭЦП в связке с удостоверяющими данными о владельце ключа, представленными в виде документа, становится сертификатом, основная функция которого заключается в подтверждении принадлежности данной ЭЦП своему владельцу.

В рамках данного семестра ставится следующая задача: реализовать систему генерации ЭЦП, способной обеспечить основные характеристики подписи: способность гарантировать целостность документа, неотказуемость авторства и само подтверждение авторства.

# 2 Обзор предметной области

Одним из российских криптопровайдеров для создания ЭЦП и проверки является КриптоПро CSP. Данный криптопровайдер соответствует требованиям ФСБ к СКЗИ для класса КС3. В нем представлен следующий функционал:

* Функции кодирования/декодирования;
* Функции работы со справочниками сертификатов;
* Высокоуровневые и низкоуровневые функции обработки криптографических сообщений.

Помимо указанных функций, КриптоПро CSP может использоваться для встраивания в прикладное программное путем непосредственного вызова функций КриптоПро CSP после загрузки модуля с использованием функции LoadLibrary.

В КриптоПро CSP реализованы следующие алгоритмы:

* Алгоритмы ЭЦП (ГОСТ Р 34.10-2012, ГОСТ Р 34.10-2001, ECDSA, RSA);
* Алгоритмы хэш-функций (ГОСТ Р 34.11-2012, ГОСТ Р 34.11-94, SHA-1, SHA-2);
* Алгоритмы шифрования (ГОСТ Р 34.12-2015 («Кузнечик» — начиная с 5.0 R2), ГОСТ 28147-89, AES (128/192/256), 3DES, 3DES-112, DES, RC2, RC4).

Разные версии данного криптопровайдера поддерживают разное количество алгоритмов и использует различные длины ключей.

Следующий продукт от того же производителя - Крипто АРМ - программа, предназначенная для шифрования и расшифрования данных, создания и проверки электронной цифровой подписи (ЭЦП) с использованием сертификатов открытых ключей, для работы с сертификатами и криптопровайдерами.

Функциональные возможности программы:

* шифрование и расшифрование файлов произвольного формата (преобразования файлов функциями СКЗИ);
* создание и проверка корректности одной или нескольких ЭЦП;
* выполнение операций подписи и шифрования за одно действие;
* управление цифровыми сертификатами и ключами пользователя, списками отозванных и доверенных сертификатов;
* управление криптопровайдерами;
* совместимость с отчуждаемыми ключевыми носителями Рутокен, eToken;
* отправка подписанных и зашифрованных файлов по e-mail.

Данная программа предоставляется в бесплатной и базовой версиях. Версия СКЗИ КриптоАРМ 5 имеет сертификаты ФСБ и соответствует требованиям ФСБ к СКЗИ для класса КС2.

Также у данного производителя имеется веб-криптопровайдер КриптоПро ЭЦП Browser plug-in, которые предназначен создания и проверки электронной подписи (ЭЦП) на веб-страницах и поддерживает работу с широким набором алгоритмов, как встроенных в операционную систему, так и доустановленных дополнительно.

Данный криптопровайдер применим в любом из современных браузеров:

* Яндекс.Браузер
* Internet Explorer
* Microsoft Edge (на базе Chromium)
* Mozilla Firefox
* Google Chrome
* Apple Safari
* Opera.

Функционал относительно ранее представленных решений значительно сокращен и предоставляет возможность подписания файлов и проверки подписи.

Остальные продукты от КриптоПро также предоставляют представленные выше функции для работы с ЭЦП, поэтому описывать подробно их функционал и комплектации далее не требуется.

# 3 Проектирование архитектуры системы

Работа клиентской части организована посредством клиентского приложения. Данное приложение осуществляет следующие функции:

* Получение команд пользователя и отображение конечного результата;
* Съем данных подписи для дальнейшей передачи серверной части;
* Генерация ключей шифрования на основе полученного сида;
* Создание сертификата стандарта PKCS1 и прочие операции с сертификатом.

Сервер представлен двумя компонентами: модуль работы с защищенным каналом обмена данных и модуль работы с нейронной сетью. Эта часть системы осуществляет следующие функции:

* Преобразование полученных данных от клиентской части системы в вид, необходимый для нейронной сети;
* Обработка данных и выдача сида, основываясь на данных о пользователе.

Система реализована как клиент-серверное приложение, архитектура которого имеет следующий вид (рисунок 3.1):

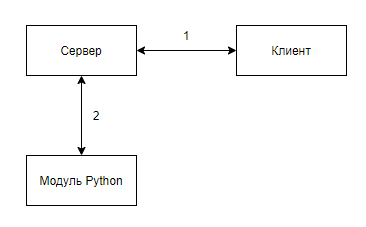


Рисунок 3.1 – Структура системы

Под 1 на данной схеме обозначено Защищенное интернет соединение между клиентской и серверной частью системы. Соединение реализует протокол TLS версии 2.1. Данное соединение необходимо для передачи данных о подписи для серверной части.

Под 2 на данной схеме обозначен защищенный именованный канал между модулем Python и основным серверным модулем.

# 4 Обоснование выбора программных средств

В качестве основного языка программирования был выбран C#, поскольку он позволяет

* Быстро реализовать оконное приложение для Windows;
* Язык обладает встроенными средствами, необходимых для реализации безопасных сетевых протоколов;
* Имеет готовые решения для реализации шифрования алгоритмом RSA и для создания сертификатов.

Для реализации нейронной сети выбран язык Python, поскольку данный язык имеет множество готовых архитектур.

# 5 Описание модулей прикладной программы

5.1 Клиентское приложение SignClient

Данное приложение предоставляет пользователям графический интерфейс для взаимодействия с системой. При старте приложение проверяет предустановленный сертификат и инициализирует TLS- соединение с серверной частью системы. После запуска и проверки сертификата открывается основное окно приложения (рисунок 5.1.1).

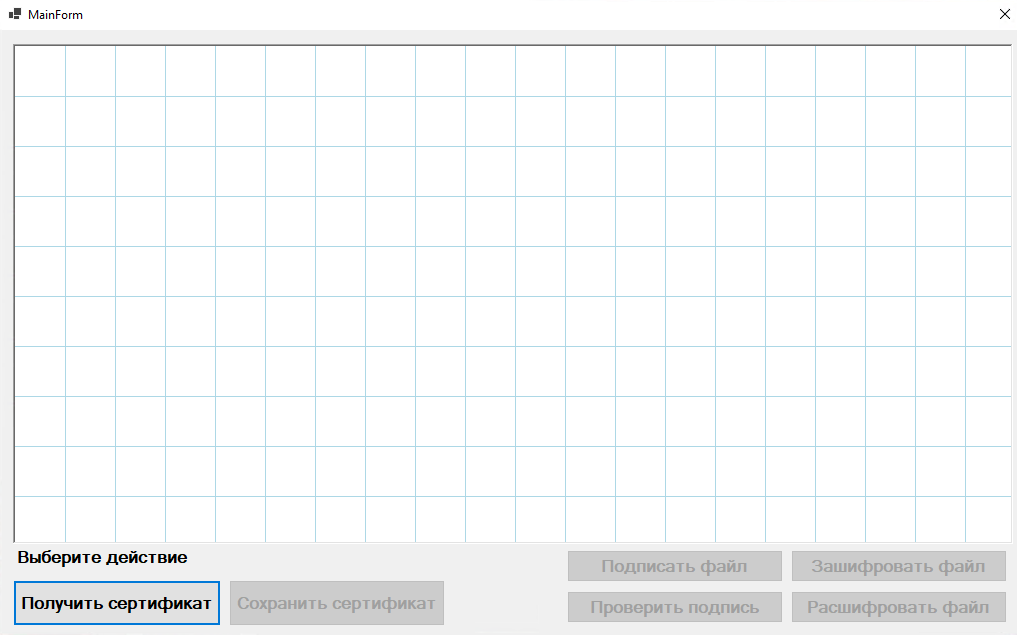


Рисунок 5.1.1 – Основное окно запуска

Далее для аутентификации требуется ввести изображение, которое было использовано для обучения нейронной сети. При правильном вводе рисунка пользователь получит доступ к остальному функционалу (рисунок 5.1.2).

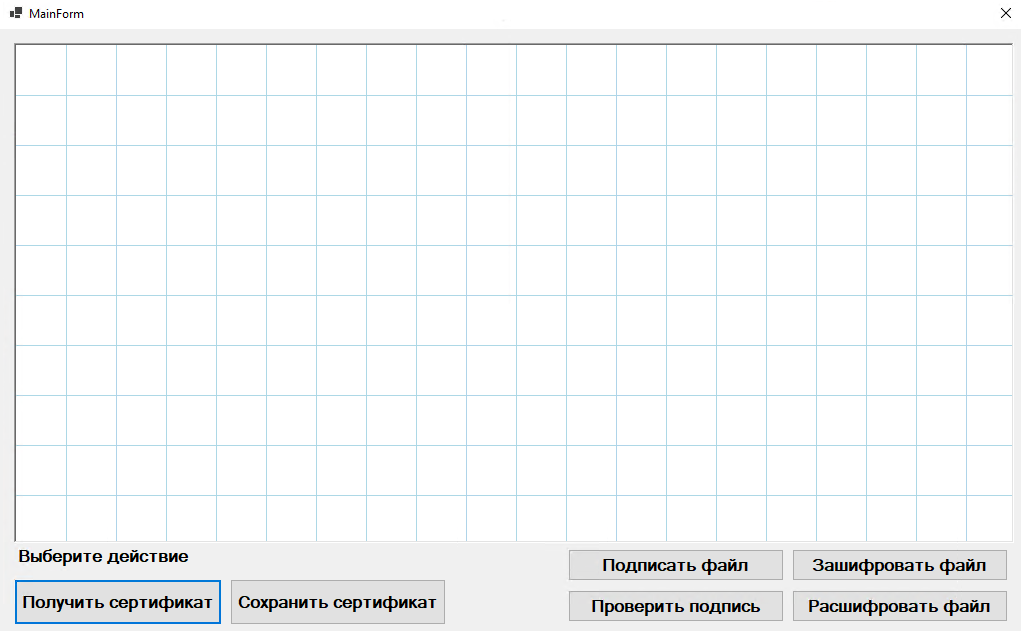


Рисунок 5.1.2 – Основное окно после аутентификации

Сохранение сертификата позволяет пользователю выбрать путь сохранения, срок окончания сертификата, имя владельца и экспортирование закрытого ключа (рисунок 5.1.3)

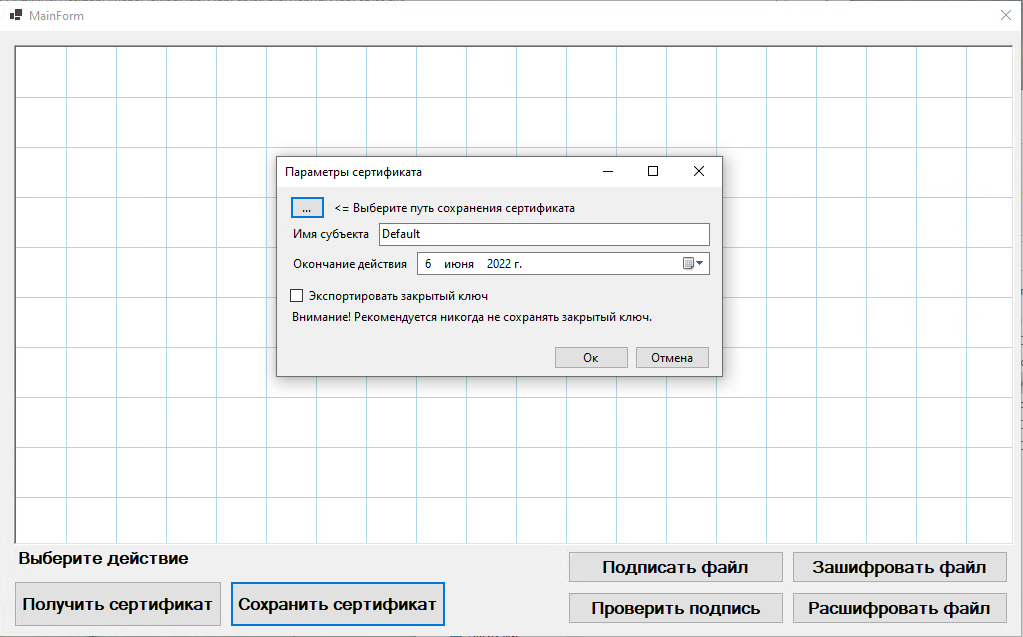


Рисунок 5.1.3 – Окно сохранения сертификата

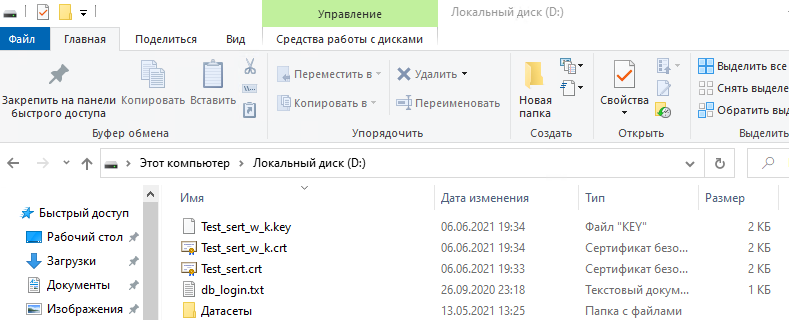


Рисунок 5.1.4 – Сохраненные сертификаты с экспортированным закрытым ключом и без

Подписать файл можно только ключом того пользователя, подписью которого был аутентифицирован пользователь. Подписание файла состоит из выбора файла и места сохранения файла (рисунки 5.1.5 – 5.1.6)

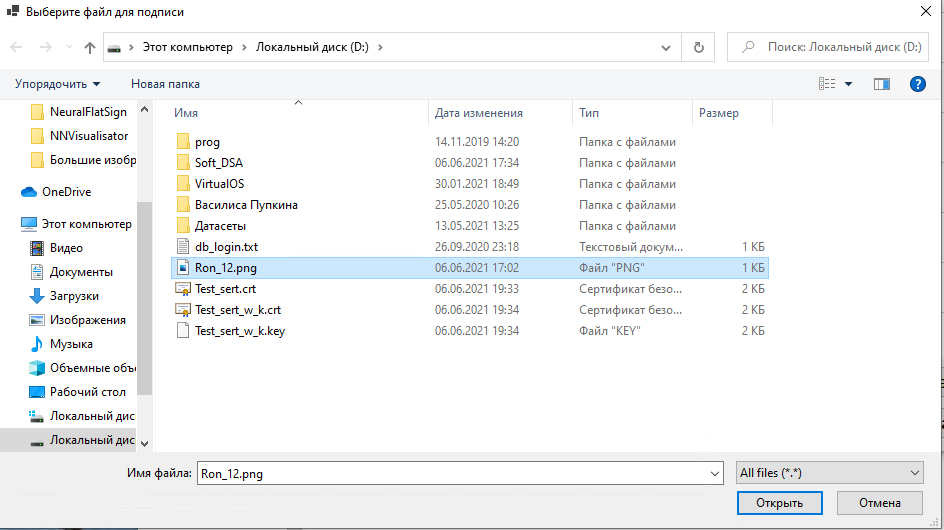


Рисунок 5.1.5 – Выбор файла для подписания

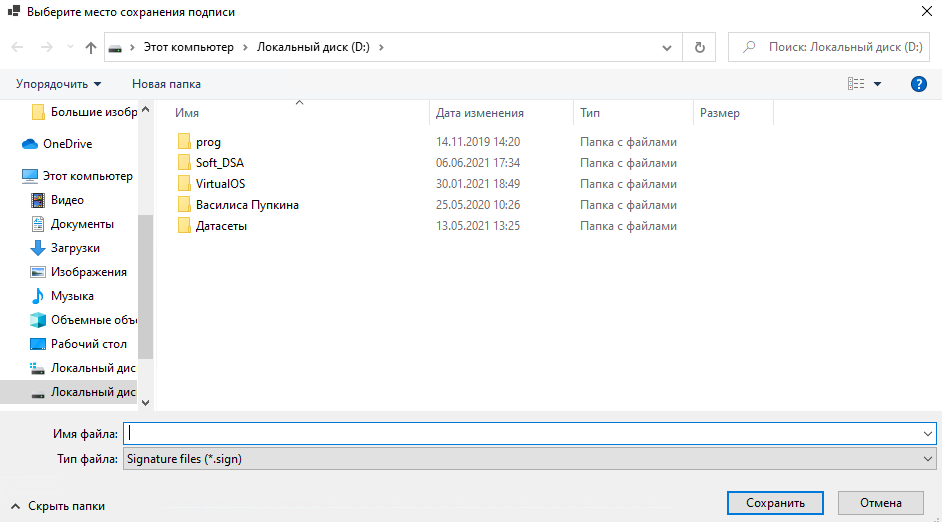


Рисунок 5.1.6 – Выбор места сохранения подписи файла и его наименования

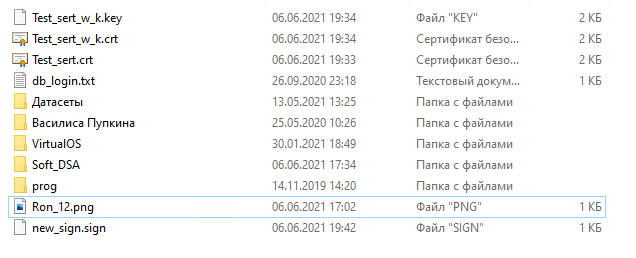


Рисунок 5.1.7 – Сохраненная подпись

Проверка подписи осуществляется выбором файла выбора файла, подписанного пользователем (рисунки 5.1.8 – 5.1.11).

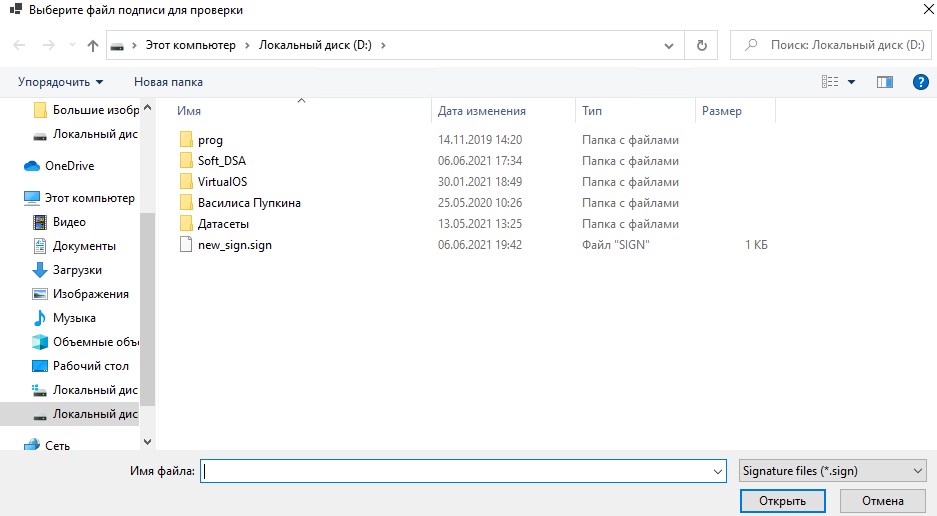


Рисунок 5.1.8 – Выбор файла подписи

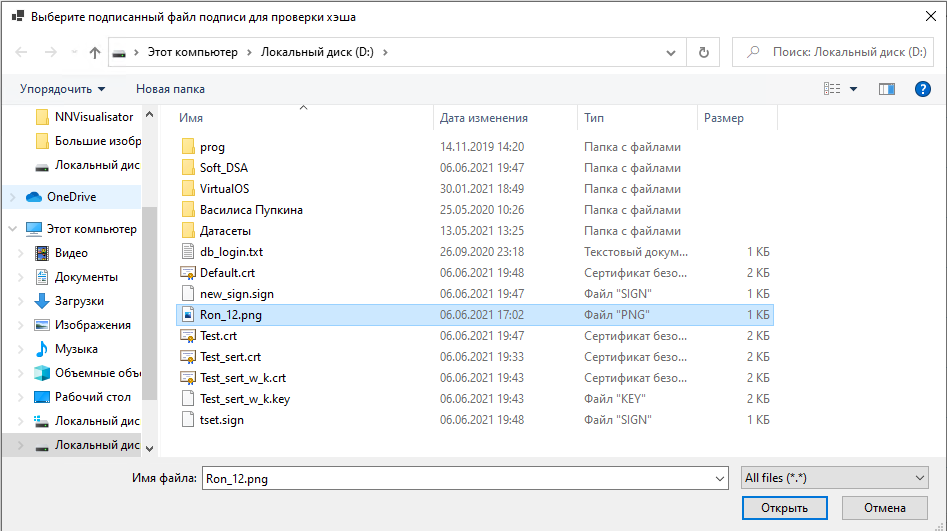


Рисунок 5.1.9 - Выбор подписанного файла

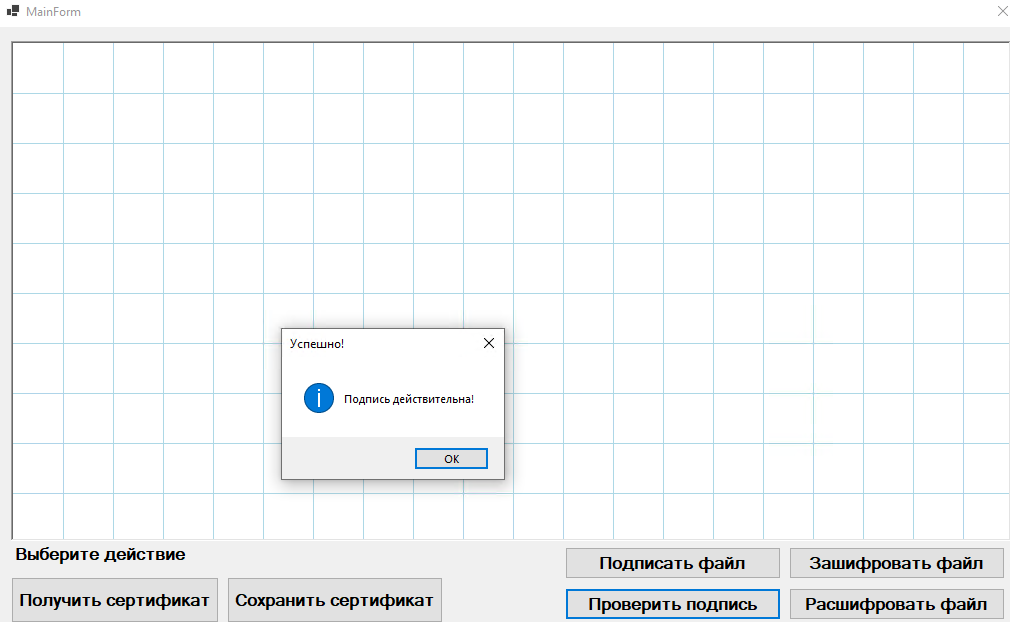


Рисунок 5.1.10 – Успешный результат проверки

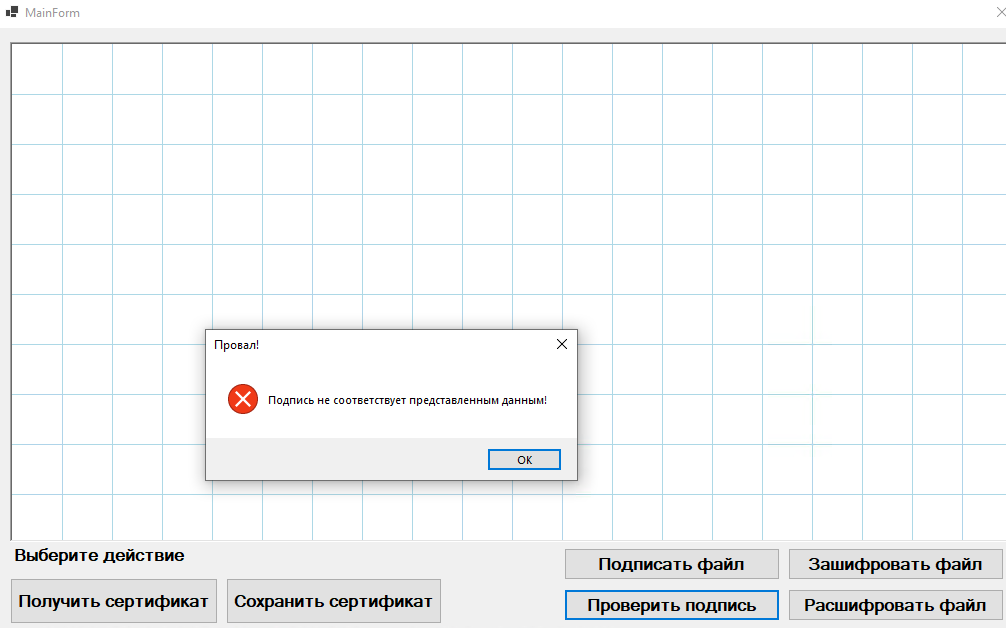


Рисунок 5.1.11 – Отрицательный результат проверки

Шифрование файла происходит следующим образом: выбор файла для шифрования, места и наименования шифрования файла (рисунки 5.1.12 – 5.1.13)

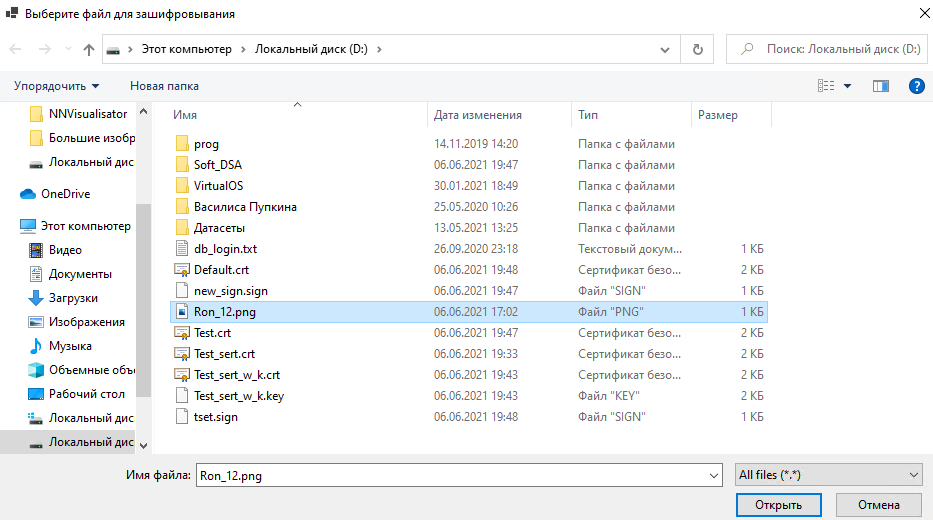


Рисунок 5.1.12 – Выбор файла для шифрования

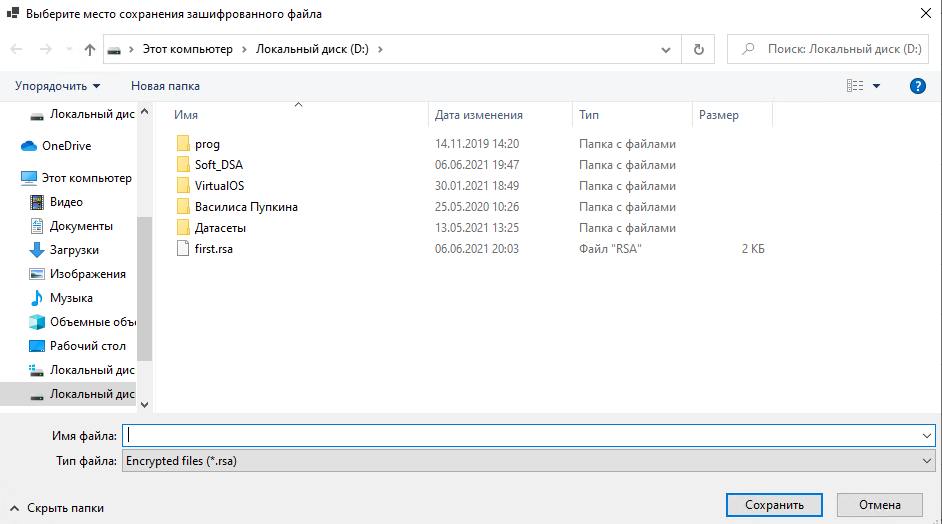


Рисунок 5.1.13 – Выбор и место для сохранения

Дешифрование файла выглядит следующим образом: выбор зашифровывания файла и место для сохранения расшифрованного файла (рисунки 5.1.14 – 5.1.)

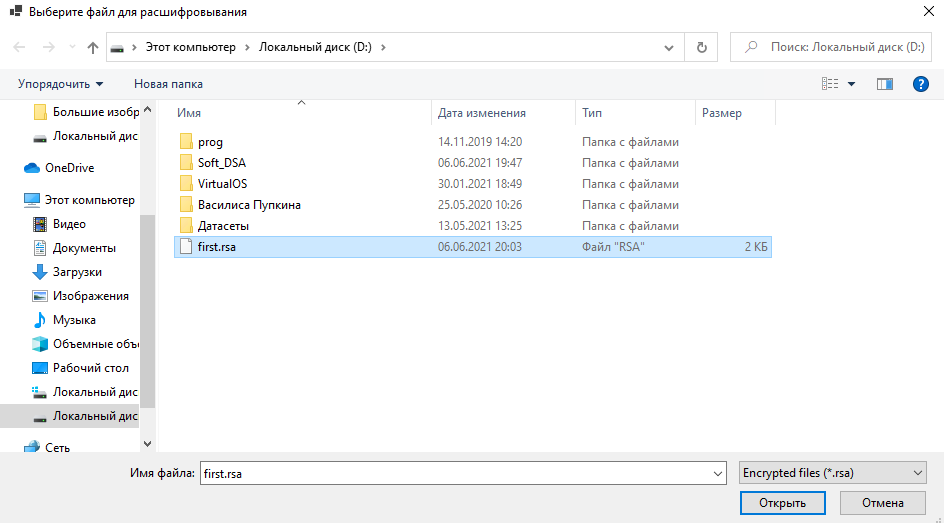


Рисунок 5.1.14 – Зашифрованный файл

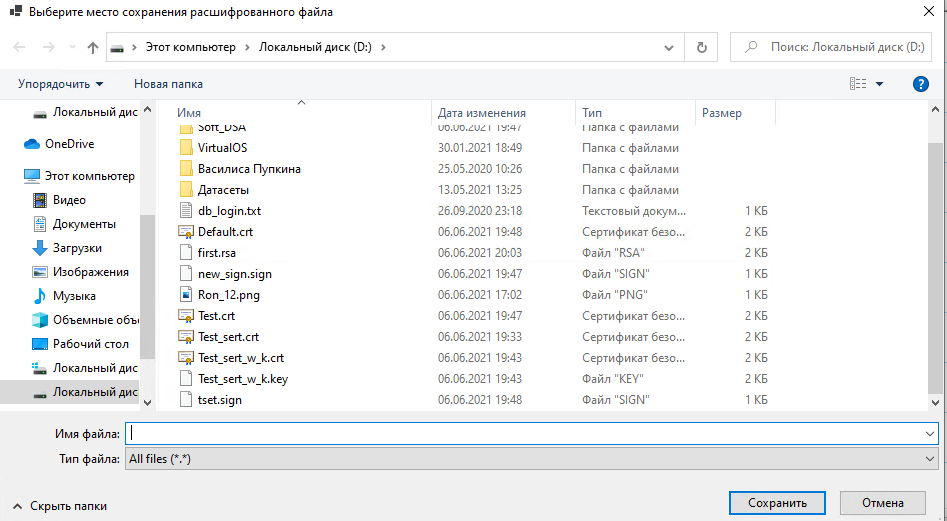


Рисунок 5.1.15 – Место для сохранения расшифрованного файла

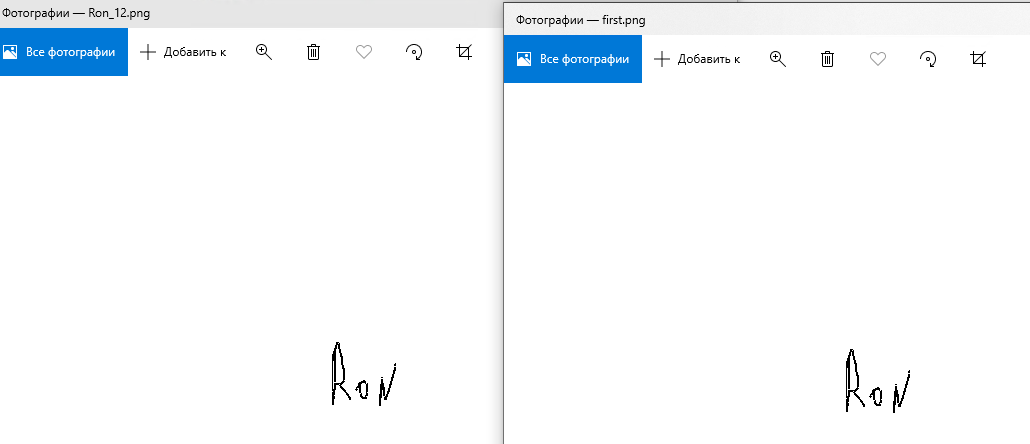


Рисунок 5.1.16 – Результат

5.2 Серверное приложение

Серверное приложение запускается на серверной машине провайдера услуг. При старте данное приложение инициализирует сокет для безопасного транспортного соединения TLS 2.1.

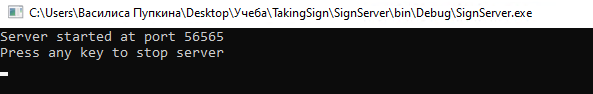


Рисунок 5.2.1 – Инициализация сокета на порту 56565

Серверное приложение представляет собой консольное приложение, регистрирующее подключение и отключение пользователей (рисунок 5.2.2), а также определяет подключенного пользователя и отправляет сид для генерации ключей шифрования на стороне клиента (рисунок 5.2.2).

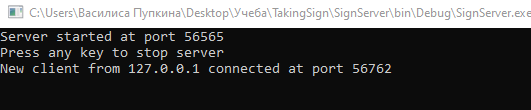


Рисунок 5.2.1 – Подключение сервера

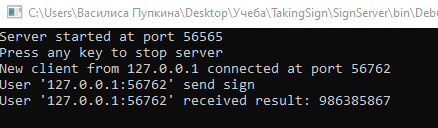


Рисунок 5.2.2 – Лог записи об отправке сида клиенту

# 6 Описание алгоритмов работы

6.1 Описание инициализации сервера

Перед инициализацией сокетов сервер осуществляет следующую подготовку:

1 При инициализации сервер создает именованный канал для обмена данными с модулем нейронной сети.

2 Сервер запускает скрипт Python для работы с нейронной сетью и ожидает подключения к именованному каналу.

3 Сервер открывает порт для подключения клиентов.

6.2 Описание протокола сетевого взаимодействия сервера и службы

В качестве транспортного протокола используется протокол TLS 2.1 в реализации C#.

Общая структура сообщений одинакова вне зависимости от требуемых действий и представляет собой код, описывающий требуемое действие или результат выполнения запроса – 1 байт и строку в формате Unicode, содержащую данные, необходимые для генерации сида пользователя.

Коды сообщений:

200 – статус успешного завершения операции;

1 – код запроса сида.

Описание взаимодействия вида клиент-сервер:

1 Клиент отправляет запрос на генерацию сида. Запрос должен быть представлен в следующем виде: 1 байт и строку в формате Unicode, содержащую данные, необходимые для генерации сида пользователя.

2 Сервер обрабатывает запрос, приводит полученные данные в вид, необходимый для нейронной сети: “хэш(<адрес клиента>\_<номер порта>)+изображение в формате png” и передает модулю нейронной сети с командой “Auth”.

3 Нейронная сеть выдает массив весов, размер которого соответствует количеству пользователей. В случае, если вес одного из пользователей превышает 0.9, то веса остальных пользователей округляется до 0, вес этого пользователя - до 1.

4 Сид вычисляется как произведение вектора сидов на вектор заранее рассчитанных случайных величин.

5 Результат произведения передается клиенту по защищённому каналу.

6.3 Описание работы клиентского приложения. Подключение к серверу и и обработка ответа

Перед инициализацией подключения приложение SignClient осуществляет проверку корректность и действительность сертификата сервера на машине пользователя. Данный сертификат используется при создании защищенного TLS –соединения.

Далее осуществляются следующие действия:

1 Считывается введенная пользователем информация

2 Подпись приводится виду, необходимому серверу для обработки: 1 байт и текст в виде координат точек.

3 Информация передается серверу по защищенному каналу TLS.

4 По полученному ответу производится генерация 2 простых чисел для сертификата стандарта PKCS1

6.4 Описание работы клиентского приложения. Генерация простыхчисел

Генерация простого 1024 битного числа осуществляется по алгоритму, прописанному в ГОСТ Р34.10-2018, модифицированного для гарантированного получения простого числа. Генерация происходит следующим образом:

1 Генерация небольшого случайного числа размером 256 бит

2 Проверка числа на четность, если четное, то прибавляется 1.

3 Проверка числа на простоту 100кратным повторение теста Ферма.

3.1 Если число не простое, то к числу прибавляется 2 и 3 пункт повторяется снова.

4 Генерируется второе рандомное нечетное число размером 256 бит.

5 Число после 3 пункта умножается на число из 4 пункта, получается 512 битное число.

5.1 Если полученное в пункте 5 число не простое, то к нему прибавляется 2.

6 Новое 512-битное число проверяется так же, как в 3 пункте.

7 Генерируется число размером 512 бит.

8 Для чисел из 7 пункта и пункта 6 повторяется перемножение, прибавление 1 в случае четности полученного произведения.

9 По аналогии с 3, 4, 5 пунктом операции повторяются, пока не получится 1024-битное простое число.

10 Весь алгоритм повторяется второй раз для генерации второго простого числа.

Сгенерированные числа передаются модулю генерации сертификата.

6.4 Описание работы клиентского приложения. Генерация простых чисел

Сертификат генерируется по стандарту PKCS1 для алгоритма RSA. Данный стандарт имеет свою специфику, потому требует генерацию 8 параметров, а именно:

* Модуль - n. Является результатом перемножения чисел из пункта 6.4 p и q.
* Открытая экспонента - e. По стандарту PRCS1 принимает значение 65537.
* Закрытая экспонента - d. Рассчитывается по формуле:

d = e-1 mod (p-1)(q-1). (1)

* Значение p. Одно из чисел из пункта 6.4.
* Значение q. Одно из чисел из пункта 6.4.
* Инвертированный q. Рассчитывается о формуле:

qin = q-1 mod p. (2)

* dp. Рассчитывается о формуле:

qin = q-1 mod p. (3)

* dq

Для корректности генерации простых чисел в качестве сида для генерации использован ответ от серверной части системы.

Поиск обратного числа по модулю реализован посредством расширенного алгоритма Евклида.

6.5 Описание работы клиентского приложения. Подпись.

Для подписи и проверки подписи используются стандартные методы RSA для С#/. Алгоритм использует 2048-битный ключ, алгоритм хэша – SHA256. Созданная подпись соответствует стандарту PKCS1.

6.6 Описание работы клиентского приложения. Шифрование

Шифрование файлов осуществляется встроенным алгоритмом RSA от С# с изменением компоновки данных:

1 Файл разбивается на блоки по 128 байт

2 Каждый блок зашифровываются алгоритмом RSA.

3 Поскольку шифрование возвращает блок от 0 до 256, то недостающие биты дозаполняются 0 до 256 бита

4 Все блоки компонуются последовательно.

6.7 Описание работы клиентского приложения. Расшифрование

Расшифрование файлов осуществляется встроенным алгоритмом RSA от С# с изменением разбиения данных:

1 Файл разбивается на блоки по 256 байт.

2 Каждый блок расшифровывается

3 Все блоки компонуются последовательно.

В результате файл будет расшифрован. При неправильном сиде файл расшифрован не будет.

# 7 Тестирование

Была протестирована возможность правильной аутентификации (рисунки 7.1 – 7.3).

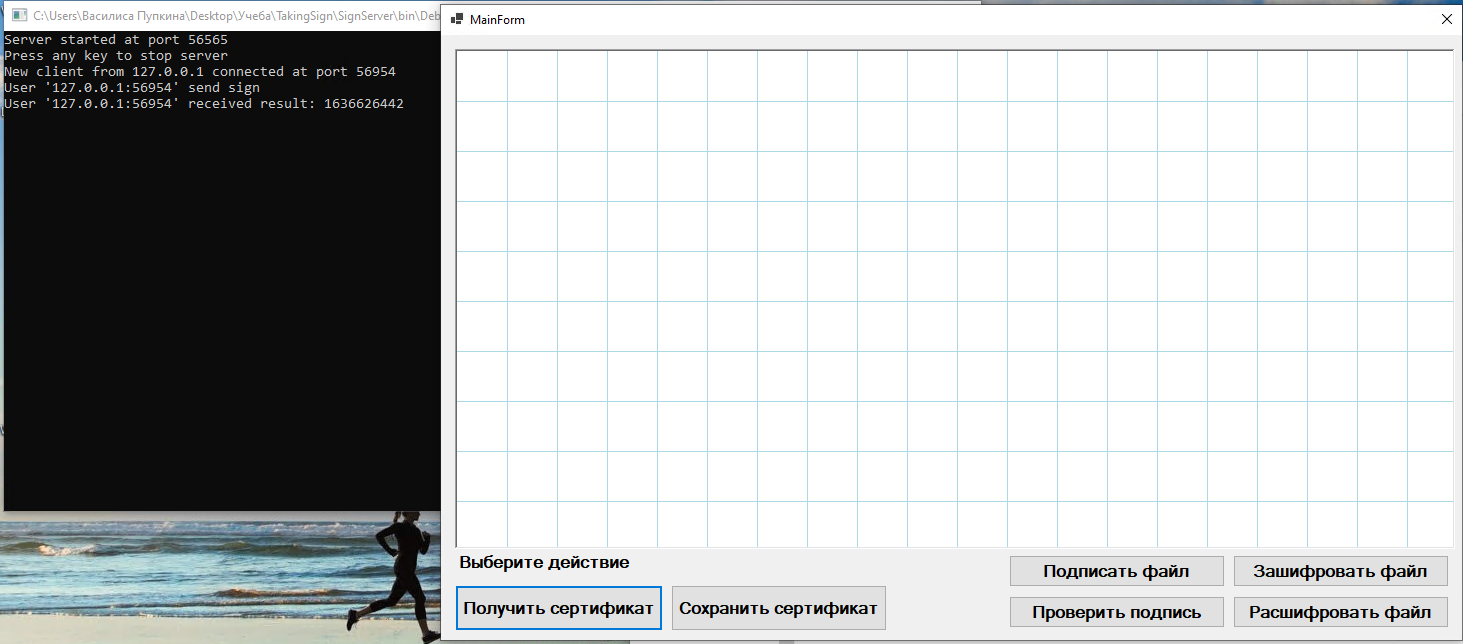


Рисунок 7.1 – Аутентификация пользователя

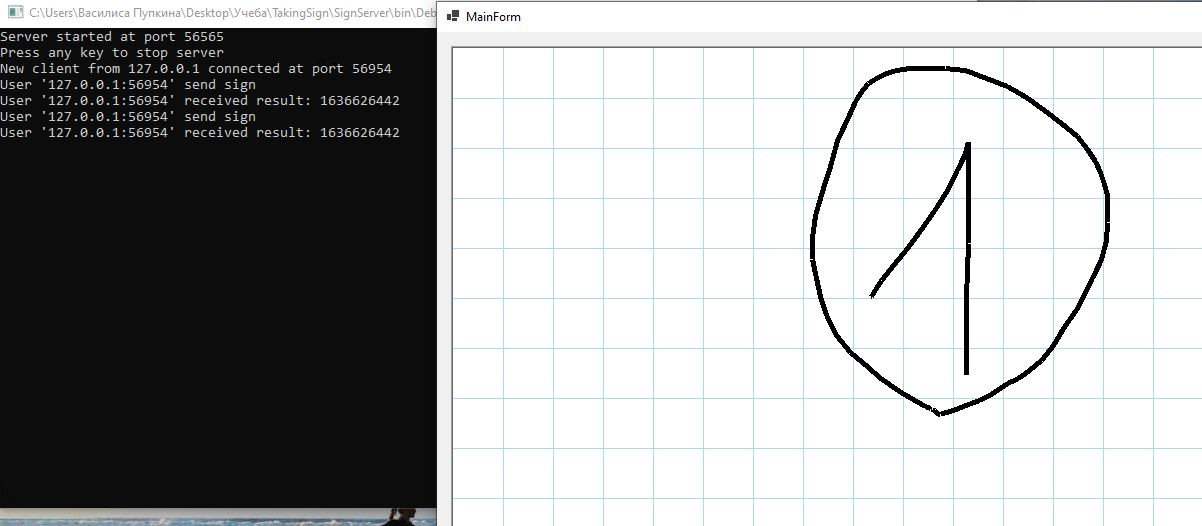


Рисунок 7.2 – Повторная аутентификация того же пользователя

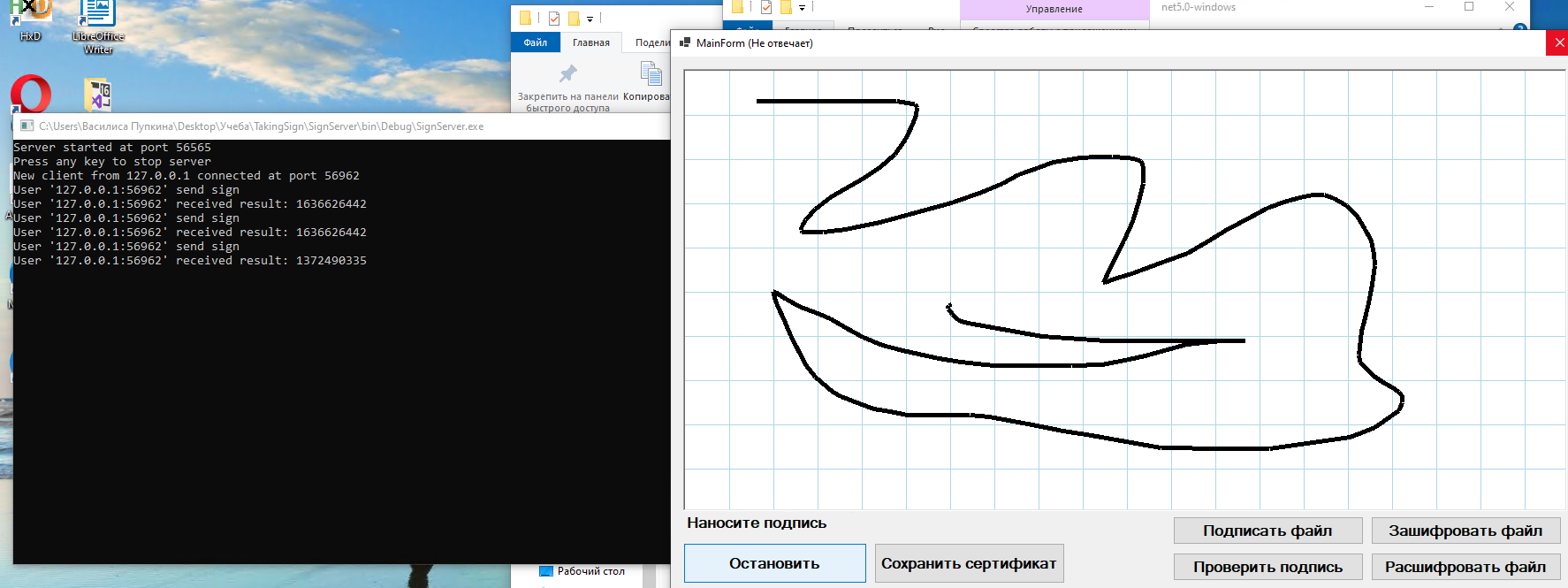


Рисунок 7.3 – Аутентификация иной подписью

Проведена проверка подписи файлов (рисунки 7.4 – 7.8).

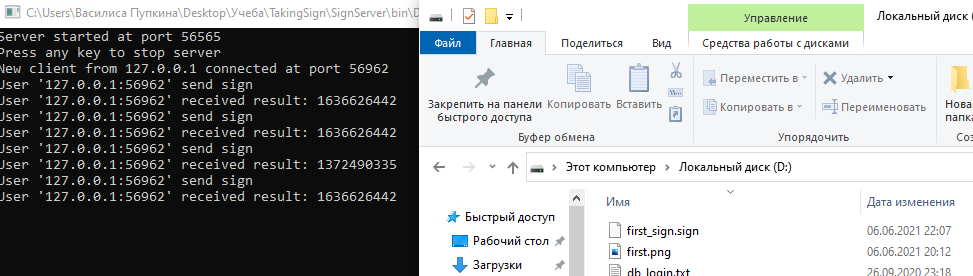


Рисунок 7.4 – Подписанный файл пользователем

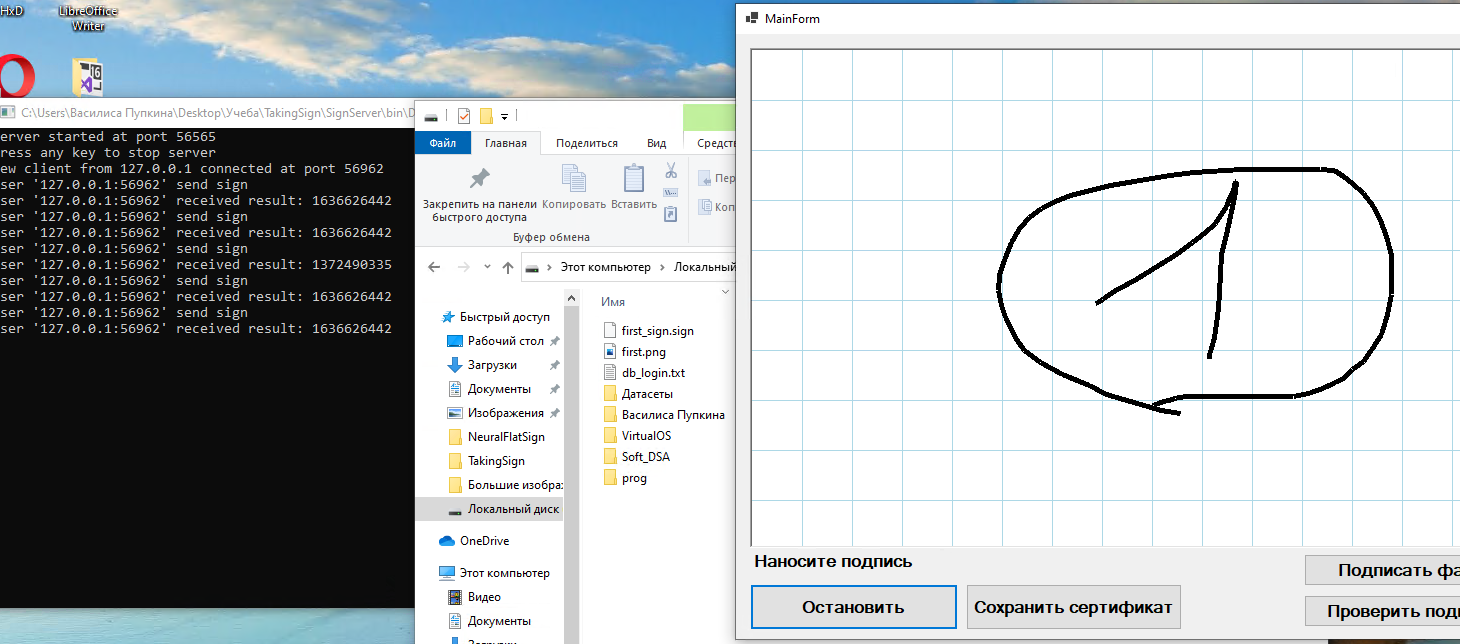


Рисунок 7.5 – Повторная аутентификация

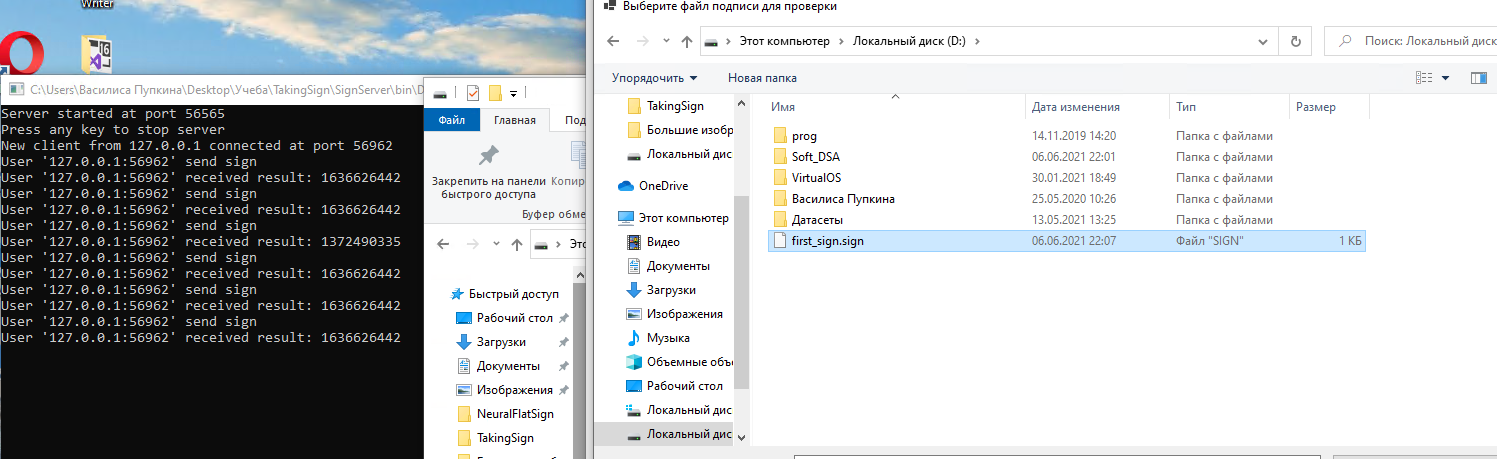


Рисунок 7.6 – Проверка подписи

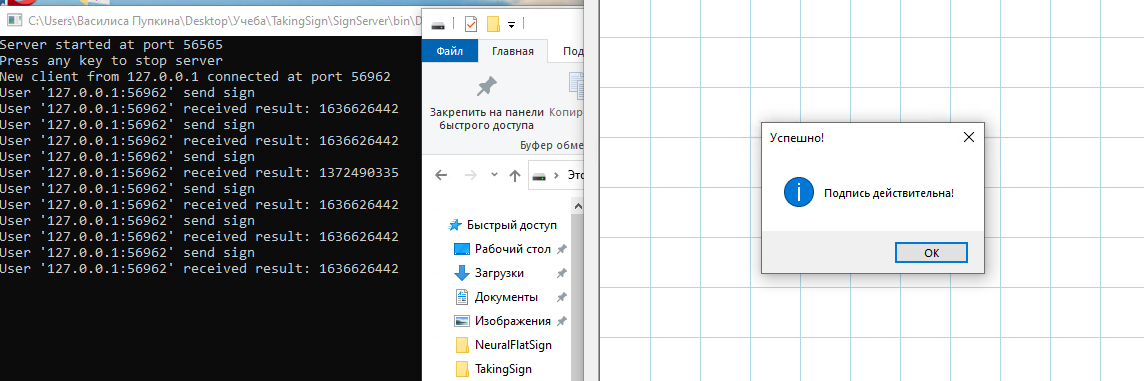


Рисунок 7.7 – Подпись верна

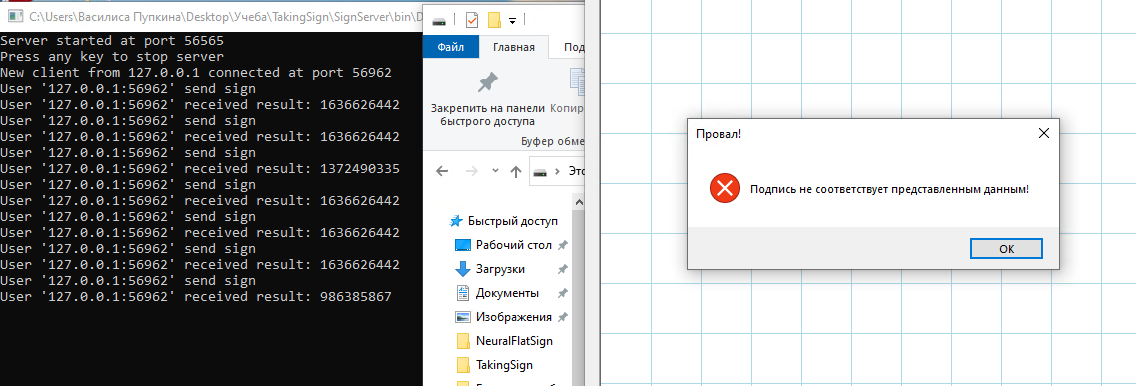


Рисунок 7.8 – Проверка подлинности подписи иным пользователем

Проведена проверка шифрования (рисунок 7.9 – 7.12).

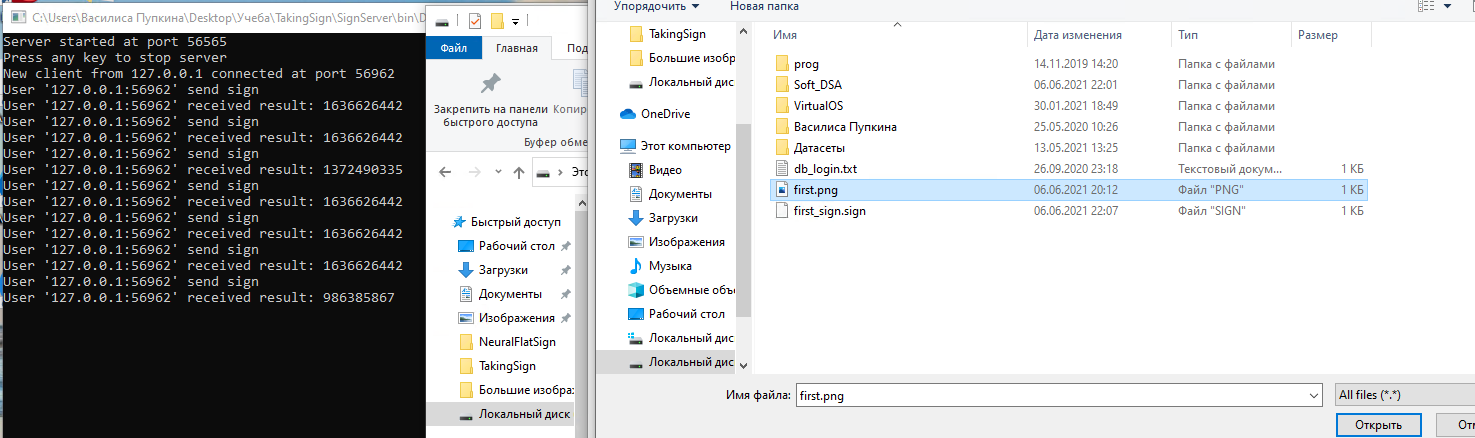


Рисунок 7.9 – Шифрование файла

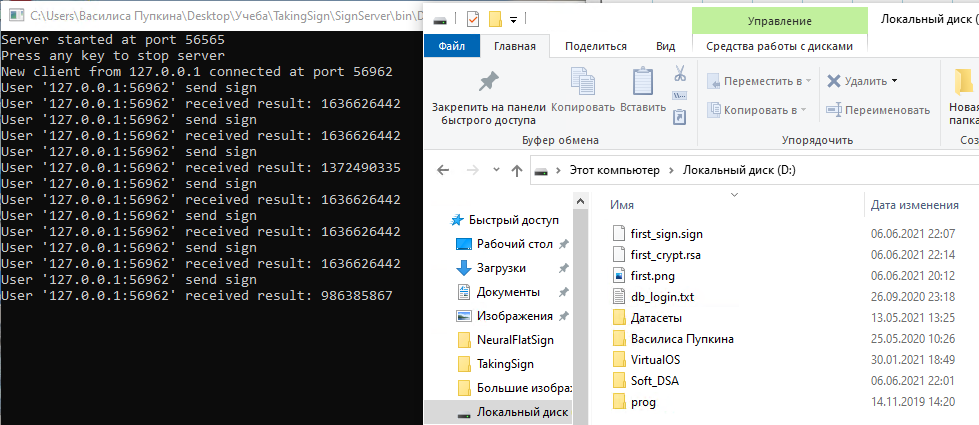


Рисунок 7.10 – Файл зашифрован

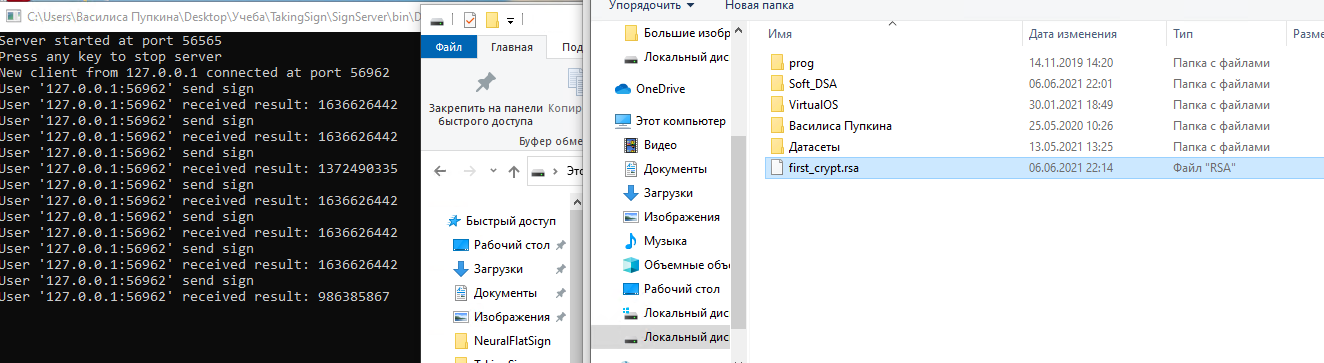


Рисунок 7.11 – Расшифрование файла пользователем, зашифровавшим файл

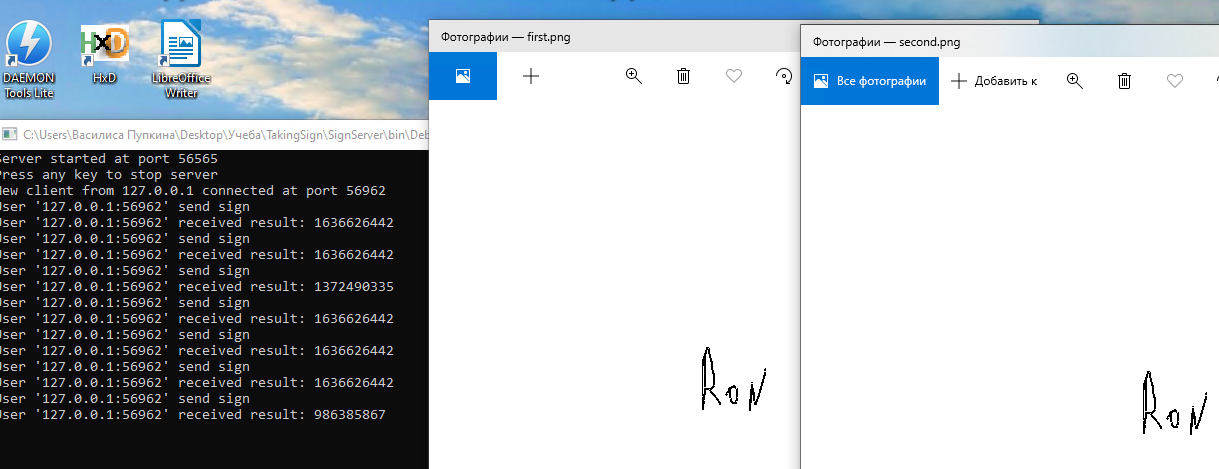


Рисунок 7.11 – Расшифрование удачно

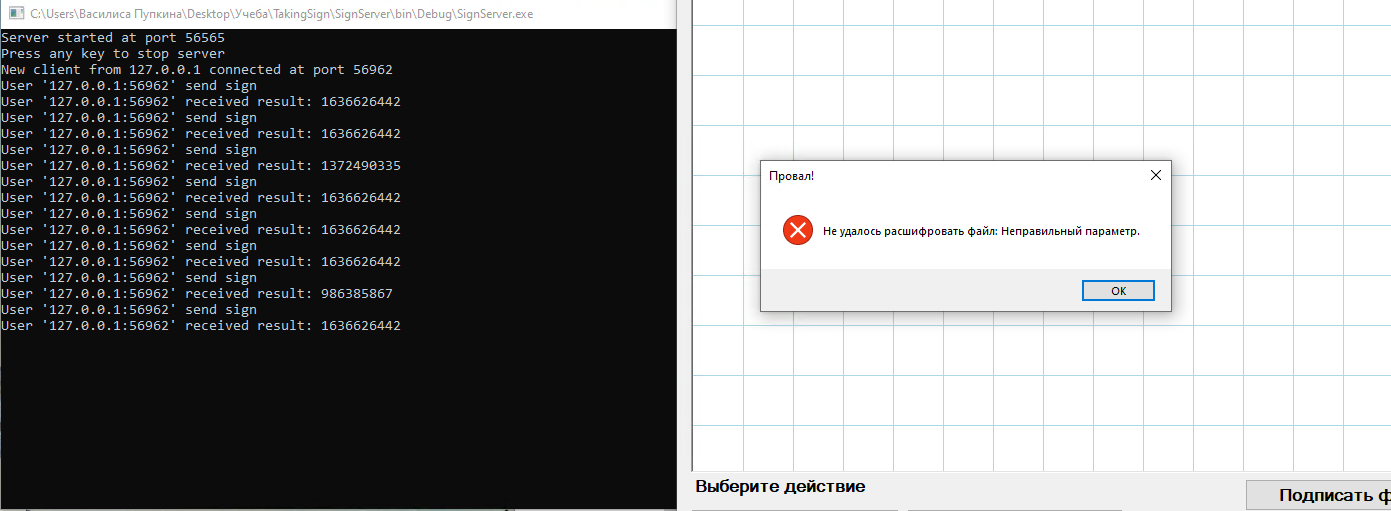


Рисунок 7.12 – Расшифрование другим пользователем

Проверка возможности подключения к серверу при нефункционирующей серверной части (рисунок 7.13).

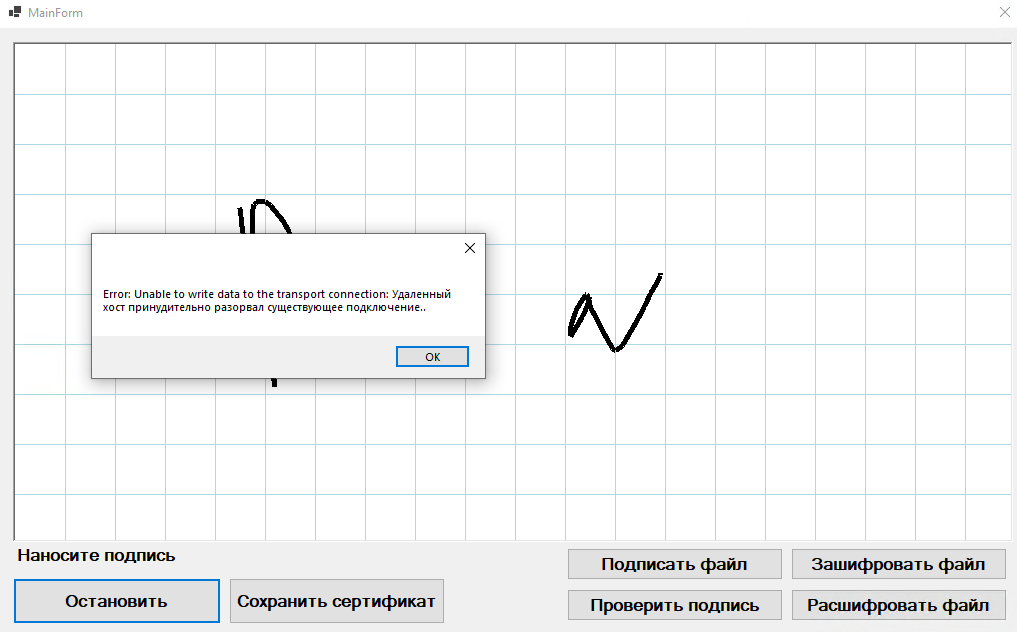


Рисунок 7.13 – Разрыв соединения

# Заключение

В ходе выполнения курсовой работы было разработано программное средство создания электронной подписи и цифровых сертификатов. Были освоены следующие навыки:

* Реализация безопасного клиент-серверного приложения;
* Межпроцессное взаимодействие программ на C# и Python;
* Работа со встроенными сертификатами стандарта PKCS1;
* Разработка алгоритма генерации больших простых чисел;
* Разработка алгоритма нахождения обратного числа по модулю.

# Список литературы

1 Руководство по программированию на C#. [Электронный ресурс]. – Режим доступа https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/ (дата обращения 01.03.2021)

2 ОС ТУСУР. [Электронный ресурс]. – Режим доступа https://storage.tusur.ru/files/40668/rules\_tech\_01-2013.pdf (дата обращения 26.05.2021)

3 Schneider M., Chang S.-F. A robust content based digital signature for image authentication // Proceedings of 3rd IEEE International Conference on Image Processing. 1996. Vol. 3. P. 227–230 vol.3.

4 Information Technology Laboratory. Digital Signature Standard (DSS): NIST FIPS 186-4. National Institute of Standards and Technology, 2013. P. NIST FIPS 186-4.

5 Федеральный закон “Об электронной подписи” от 06.04.2011 N 63-ФЗ (последняя редакция) / КонсультантПлюс [Электронный ресурс]. URL: http://www.consultant.ru/document/cons\_doc\_LAW\_112701/ (дата обращения: 21.05.2021).

6 КриптоПро | Использование КриптоПро CSP [Электронный ресурс]. URL: https://www.cryptopro.ru/products/csp/usage (дата обращения: 31.05.2021).

7 ГОСТ 34.10-2018 Информационная технология (ИТ). Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи (с Поправкой) - docs.cntd.ru [Электронный ресурс]. URL: https://docs.cntd.ru/document/1200161706 (дата обращения: 10.03.2021).

8 RSA Cybersecurity and Digital Risk Management [Электронный ресурс] // RSA.com. URL: https://www.rsa.com (дата обращения: 30.03.2021).

Приложение А

(обязательное)

Руководство пользователя

1 Введение

1.1 Область применения

Требования настоящего документа применяются при эксплуатации.

1.2 Краткое описание возможностей

Система предназначена для работы с ЭЦП и сертификатами, сгенерированных на основе данных изображений подписи пользователей. Для удобства эксплуатации пользователям предоставляется клиентское приложение, предоставляющее управление основным функционалом системы.

1.3 Уровень подготовки пользователя

Пользователь должен иметь опыт работы с:

- операционными системами семейства Windows;

- оконными приложениями Windows.

1.4. Перечень эксплуатационной документации

Руководство пользователя.

2 Назначение и условия применения

2.1 Виды деятельности, функции

Клиентское приложение позволяет генерировать на основе изображения подписи ключи шифрования и сертификат с данными о пользователе и сгенерированном ключе. Также приложение позволяет подписывать выбранные пользователем файлы, проверять подпись, шифровать файлы и дешифровать.

Также приложение позволяет экспортировать данные ключа в виде сертификата с экспортом закрытого ключа и без него.

2.2 Программные и аппаратные требования к системе

Со стороны пользователя требования к аппаратуре следующие:

* процессор на архитектуре x86-64;
* ОС Windows 7 с пакетом обновления 1 и выше;
* .NET Core 5.0 или выше;
* наличие интернет соединения.

3 Работа с системой

3.1 Запуск системы

1 Для того, чтобы запустить клиент SignClient, запустите соответствующий ему исполняемый файл;

2 Получите сертификат, введя данные подписи.

3.2 Проверка работоспособности системы

Проверка работоспособности приложения и доступности сервиса TakingSign сводится к следующему:

* Запуск приложения;
* Экспорт сертификата;
* Выход из системы и повторный экспорт сертификата;
* Сверка данных экспортированных сертификатов на предмет соответствия и корректности аутентификации.

Возможна проверка работоспособности приложения и доступности сервиса TakingSign иным способом:

* Запуск приложения;
* Шифрование любого файла;
* Выход из системы и повторная аутентификация;
* Дешифрование ранее зашифрованного файла;
* При неудаче – повторение аутентификации до положительного результата.

4 Описание операций

4.1 Наименование операции

Клиентское приложение SignClient выполняет следующие функции.

Таблица 4.1 – Функции клиентского приложения

|  |  |
| --- | --- |
| Функции | Задачи |
| Взаимодействие клиента с сервером | Аутентификация клиента |
| Получение от сервера секрета для генерации ключей шифрования |
| Сохранение сертификата | Создание сертификата с экспортом закрытого ключа и без |
| Работа с файлами | Подписание файла |
| Проверка подписи |
| Шифрование файла |
| Дешифрование файла |

4.2 Условие выполнения операций

Задача «Аутентификация клиента» требует следующих условий для выполнения:

* Подключение к интернету;
* Наличие функционирующей серверной части системы.

Подготовительные действия:

* Запуск клиентского приложения

Основные действия:

* Нажать кнопку «Получить сертификат»;
* Ввод данных подписи;
* Нажать кнопку «Остановить».

Заключительные действия: не требуются

Задача «Получение от сервера секрета для генерации ключей шифрования» требует следующих условий для выполнения:

* Подключение к интернету;
* Наличие функционирующей серверной части системы.

Подготовительные действия:

* Выполнение задачи «Аутентификация клиента»

Основные действия: не требуются

Заключительные действия: не требуются

Задача «Создание сертификата с экспортом закрытого ключа и без» требует следующих условий для выполнения:

* Выполнение задачи «Получение от сервера секрета для генерации ключей шифрования»

Подготовительные действия: не требуются

Основные действия:

* Нажать кнопку «Сохранить сертификат»;
* Выбрать путь сохранения, имея субъекта, срок действия и параметр экспортирования закрытого ключа.

Заключительные действия: не требуются

Задача «Подписание файла» требует следующих условий для выполнения:

* Выполнение задачи «Получение от сервера секрета для генерации ключей шифрования»

Подготовительные действия: не требуются

Основные действия:

* Нажать кнопку «Подписать файл»;
* Выбрать файл для подписания, путь сохранения файла;
* Ввод наименования файла подписи.

Заключительные действия: нажать кнопку «Ок» во всплывающем окне

Задача «Проверка подписи» требует следующих условий для выполнения:

* Выполнение задачи «Получение от сервера секрета для генерации ключей шифрования»;
* Наличие подписанного файла и исходного файла

Подготовительные действия: не требуются

Основные действия:

* Нажать кнопку «Проверить подпись»;
* Выбрать файл подписи;
* Ввод подписанный файл.

Заключительные действия: нажать кнопку «Ок» во всплывающем окне.

Задача «Шифрование файла» требует следующих условий для выполнения:

* Выполнение задачи «Получение от сервера секрета для генерации ключей шифрования»;
* Наличие исходного файла

Подготовительные действия: не требуются

Основные действия:

* Нажать кнопку «Зашифровать файл»;
* Выбрать файл для шифрования;
* Выбрать место сохранения зашифрованного файла и его наименование.

Заключительные действия: нажать кнопку «Ок» во всплывающем окне

Задача «Расшифрование файла» требует следующих условий для выполнения:

* Выполнение задачи «Получение от сервера секрета для генерации ключей шифрования»;
* Наличие исходного зашифрованного файла

Подготовительные действия: не требуются

Основные действия:

* Нажать кнопку «Расшифровать файл»;
* Выбрать файл для дешифрования;
* Выбрать место сохранения расшифрованного файла и его наименование.

Заключительные действия: нажать кнопку «Ок» во всплывающем окне

5 Аварийные ситуации

В ходе работы с приложением возможны представленные в таблице 5.1 аварийные ситуации.

Таблица 5.1 Аварийные ситуации при работе с приложением TakingSign

|  |  |  |  |
| --- | --- | --- | --- |
| Класс ошибки | Ошибка | Описание ошибки | Требуемые действия пользователя при возникновении ошибки |
| Ошибка аутентификации | Получение несоответствующего пользователю сида | Веденные пользователем данные были недостаточно точными для аутентификации | Проверка данных аутентификации |
| Ошибка работы сервера | Отсутствие ответа сервера | Сервер разорвал соединение ввиду ошибки на стороне сервера | Отсутствуют |

6 Рекомендации по освоению

Для успешного освоения клиентского приложения TakingSign необходимо иметь навыки работы с вычислительными устройствами и изучить следующее:

- Настоящее «Руководство пользователя».

Приложение Б

(обязательное)

Руководство администратора

1 Условия применения программы

1.1. Требования к техническим средствам

Требования к техническим средствам серверной части системы программное средство создания электронной подписи и цифровых сертификатов следующие:

* процессор x86-64;
* оперативная память – не менее 1 Гб;
* свободного места на накопителе не менее 100 Мб;
* наличие интернет соединения.

1.2. Требования к общему программному обеспечению (ОПО)

Требования к общему программному обеспечению (ОПО):

* операционная система Windows 7 с пакетом обновления 1 или выше или Windows Server 2008 R2 с пакетом обновления 1 или выше; Microsoft .NET Framework 4.7.2 или выше;
* Python 3.8.2 с установленными модулями Keras 2.3.1, tensorflow 2.2.0rc3, pillow (любой версии), numpy 1.9.1.

2 Характеристики программы

Система рассчитана на круглосуточный режим работы в течение 365 дней в году. Средствами контроля и мониторинга являются автоматизированные встроенные в платформу программные компоненты Серверное приложение представляет собой консольное приложение, содержащее в себе следующие инструменты:

1 Инструменты для мониторинга подключенных пользователей и вывода уникального кода подключенного пользователя.

2 Консоль с выводом событий, произошедших на сервере, для удобного администрирования.

3 Сообщения

Во время работы приложения или взаимодействия с ним администратор может получать сообщения.

При старте сервера в лог выводятся следующие сообщения:

«Server started at port 56565

Press any key to stop server»

«New client from <адрес пользователя> connected at port 57088 – подключение нового клиента. До отключения пользователя связка адрес пользователя и сокета будет использоваться как идентификатор пользователя в логах.

«Error, when accept new client» - Произошла ошибка сервера при подключении нового клиента.

«Failed to add new client with error: <описание ошибки>» - Не удалось добавить нового клиента.

«Error when remove dead client: <описание ошибки>» - Не удалось удалить структуры, связанные с отключившимся пользователем.

«Error at user '<идентификатор>' <описание ошибки>» - Произошла сетевая ошибка или пользователь был отключен.

«User '<идентификатор>' send sign – пользователь отправил серверу запрос на выдачу сида генерации

User '<идентификатор>' received result: <сид> - пользователю отправлен результат.