**Лабораторная работа 22**

**ПСКП, ПОИТ-3**

1. Для выполнения лабораторной работы, студенты должны работать по парам, поочередно выполняя роли **CA** (Certification authority) и **Resource**.
2. В создаваемых в процессе выполнения лабораторной работы данных будет использоваться обозначения **XYZ, ABC** – инициалы студентов (IPN – Иванов Петр Николаевич).
3. Для генерации сертификатов примените библиотеку **OpenSSL**.

**Задание 01 CA**

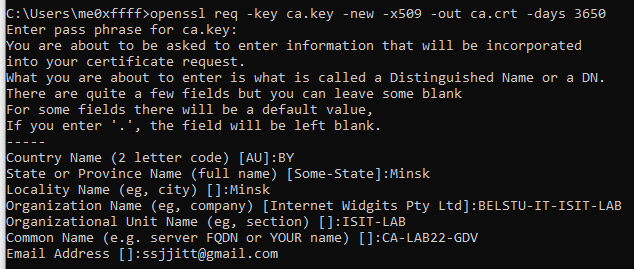
Сперва бацаем ключик **ca.key**.



strwplab везде

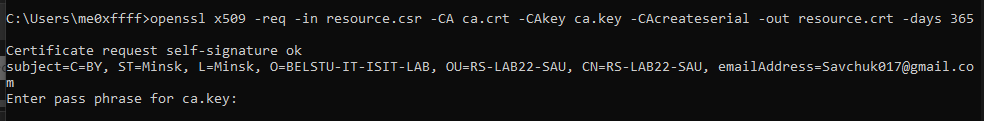
1. Создайте сертификат CA с CN = **CA-LAB22-XYZ**.

**ca.crt**



1. Получите от **Resource** запрос на генерацию сертификата.

Предварительно Resource должен предоставить запрос на сертификат. Этот запрос будет в виде файла **resource.csr.**



1. Сгенерируйте сертификат для **Resource** позаданному им запросу. Разрешите доступ к следующим доменам: **LAB22-ABC** и **ABC**



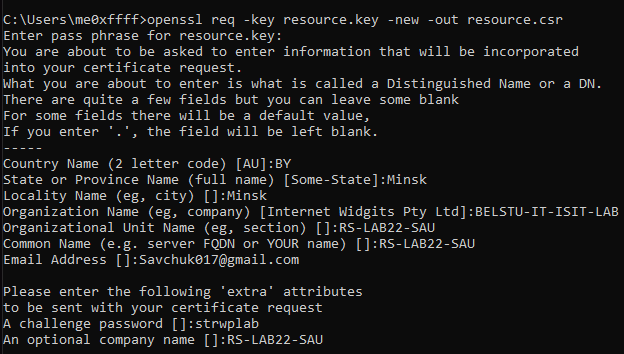
1. Передайте **Resource** сгенерированный сертификат и свой CA- сертификат.

Через телегу перекинуть короче два файла: resource.crt и ca.crt.

**Задание 02 Resource**



1. Сгенерируйте запрос на сертификат к CA с CN = **RS-LAB22- ABC** и передайте его CA для генерации сертификата.



1. Получите от СА два сертификата.

Тг

1. Импортируйте СА-сертификат в хранилище сертификатов (доверенные центры сертификации).

Win+r -> mmc -> добавить или удалить оснастку -> сертификаты -> добавить для текущий пользователь -> доверенные корневые центры сертификации -> импортировать ca.crt

1. Разработайте приложение **22-01**, которое принимает GET-запросы по https-протоколу. Тестирование выполните с помощью браузера.

Resource запускает сервер, клиент коннектит localhost

**Задание 03 CA/Resource**

1. Поменяйтесь dв паре ролями и выполните лабораторную работу с другой ролью.

Хорошо, давай по шагам разберем, как выполнить лабораторную работу. Я пошагово объясню, что делать на каждом этапе. Будем использовать **OpenSSL** для создания сертификатов и **Node.js** для разработки HTTPS-сервера.

**Задание 01: Работа с CA (Certification Authority)**

**Шаг 1: Создание приватного ключа для CA**

1. **Создай каталог для работы с сертификатами**: Это поможет не потерять файлы. Например, можно создать папку C:\certs.

Открой командную строку и выполни:

mkdir C:\certs

cd C:\certs

1. **Создание приватного ключа для CA**: Для создания приватного ключа удостоверяющего центра (CA) используй команду:
2. openssl genpkey -algorithm RSA -out ca.key -aes256

Это создаст файл ca.key, который будет защищен паролем.

**Шаг 2: Создание сертификата CA**

1. **Создание самоподписанного сертификата для CA**: После создания ключа нужно создать самоподписанный сертификат для CA с Common Name (CN) = CA-LAB22-XYZ:
2. openssl req -key ca.key -new -x509 -out ca.crt -days 3650

Это создаст сертификат ca.crt, который будет использоваться для подписи сертификатов для Resource.

**Шаг 3: Получение запроса на сертификат от Resource**

1. **Получение CSR (Certificate Signing Request) от Resource**: Resource должен предоставить тебе запрос на сертификат. Этот запрос будет в виде файла resource.csr. Убедись, что этот файл у тебя есть. Он будет создан в следующем шаге на стороне Resource.

**Шаг 4: Генерация сертификата для Resource**

1. **Подписание запроса от Resource**: Используя приватный ключ CA (ca.key) и сертификат CA (ca.crt), подписываем запрос на сертификат от Resource:
2. openssl x509 -req -in resource.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out resource.crt -days 365
3. **Разрешение доступа к доменам**: В процессе создания сертификата для Resource можно указать домены, к которым будет разрешен доступ. Для этого нужно использовать расширения в конфигурационном файле OpenSSL или вручную добавить их через команду:
4. openssl req -new -key resource.key -out resource.csr -subj "/CN=RS-LAB22-ABC"

Домену нужно будет разрешить доступ к LAB22-ABC и ABC в конфигурации сертификата.

**Шаг 5: Передача сертификатов Resource**

1. После генерации сертификата для Resource, передай ему два файла:
   * resource.crt — сертификат Resource.
   * ca.crt — сертификат CA (он будет использоваться для проверки доверия к сертификату).

**Задание 02: Работа с Resource**

**Шаг 6: Генерация запроса на сертификат от Resource**

1. **Создание приватного ключа для Resource**: Resource должен создать свой приватный ключ:
2. openssl genpkey -algorithm RSA -out resource.key -aes256
3. **Создание запроса на сертификат (CSR)**: Теперь Resource создает запрос на сертификат с Common Name (CN) = RS-LAB22-ABC:
4. openssl req -key resource.key -new -out resource.csr

**Шаг 7: Получение сертификатов от CA**

1. **Получение сертификатов от CA**: Resource получает два сертификата:
   * Сертификат от CA (ca.crt).
   * Собственный сертификат (resource.crt), который подписал CA.

**Шаг 8: Импорт сертификата CA в хранилище доверенных сертификатов**

1. **Импорт CA-сертификата в хранилище сертификатов**: На Windows открой **MMC консоль** и добавь оснастку **Сертификаты**:
   * Запусти mmc, выбери **Файл** -> **Добавить или удалить оснастку**.
   * Выбери **Сертификаты** и добавь его для **Текущий пользователь**.
   * Перейди в раздел **Доверенные корневые центры сертификации**.
   * Импортируй файл **ca.crt** (сертификат удостоверяющего центра).

**Шаг 9: Разработка приложения для HTTPS**

1. **Создание простого HTTPS-сервера на Node.js**: Пример простого кода на Node.js для создания HTTPS-сервера:

Сначала установи Node.js с [официального сайта](https://nodejs.org/).

1. **Создание файла app.js**: Открой текстовый редактор и создай файл app.js с таким содержанием:
2. const fs = require('fs');
3. const https = require('https');
4. const options = {
5. cert: fs.readFileSync('resource.crt'), // Путь к сертификату Resource
6. key: fs.readFileSync('resource.key'), // Путь к приватному ключу Resource
7. ca: fs.readFileSync('ca.crt') // Путь к сертификату CA
8. };
9. https.createServer(options, (req, res) => {
10. res.writeHead(200);
11. res.end('Hello, this is a secure connection over HTTPS!');
12. }).listen(5000, () => {
13. console.log('HTTPS server is running on https://localhost:5000');
14. });
15. **Запуск сервера**: В командной строке перейди в папку с файлом app.js и запусти:
16. node app.js
17. **Тестирование**: Перейди в браузер и введи:
18. https://localhost:5000

Браузер должен отобразить сообщение: **"Hello, this is a secure connection over HTTPS!"**.

**Задание 03: Смена ролей**

Когда ты и твой партнер выполните оба эти задания, поменяйтесь ролями и повторите работу для другой роли.

Теперь ты готов к выполнению лабораторной работы. Если возникнут вопросы или что-то не получится, обязательно обращайся!