**HTTPS (HyperText Transfer Protocol Secure).** Это расширение протокола HTTP, которое поддерживает шифрование.

Чтобы сайт работал по HTTPS, ему **необходим SSL/TLS сертификат**.

**SSL/TLS (Secure Sockets Layer / Transport Layer Security).** Это криптографические протоколы, которые обеспечивают защищенную передачу данных. HTTPS использует SSL/TLS для шифрования. SSL – это более старая версия, TLS – более новая и безопасная.

**Приватный ключ**: Секретный. Хранишь у себя, никому не показываешь. Используется для:

* Расшифровки данных, зашифрованных твоим публичным ключом.
* Создания цифровой подписи (доказательство, что это ты).

**Публичный ключ**: Не секретный. Можно раздавать всем. Используется для:

* Шифрования данных, которые сможет расшифровать только владелец приватного ключа.

**CA (Certification Authority – Центр Сертификации):** Это доверенная третья сторона, которая выпускает (подписывает) цифровые сертификаты.

**Корневой сертификат CA (Root CA Certificate):** Это самоподписанный сертификат. CA сам удостоверяет свой собственный публичный ключ. Этот корневой сертификат должен быть установлен в хранилище доверенных сертификатов на клиентских машинах, чтобы они доверяли сертификатам, выданным этим CA.

**CSR (Certificate Signing Request – Запрос на Подпись Сертификата).** Это файл, который генерирует тот, кто хочет получить сертификат (например, владелец веб-сервера).

* **Что содержит CSR:**
  + Информацию о заявителе (CN, организация, страна и т.д.).
  + Публичный ключ заявителя.
* Заявитель подписывает CSR своим приватным ключом (неявно, при генерации).
* Этот файл отправляется в CA. CA проверяет информацию и, если все в порядке, использует данные из CSR для создания и подписи сертификата.

**OpenSSL:** Это мощная библиотека и набор утилит командной строки с открытым исходным кодом для работы с SSL/TLS. Позволяет генерировать ключи, CSR, сертификаты, подписывать их, конвертировать форматы и многое другое.

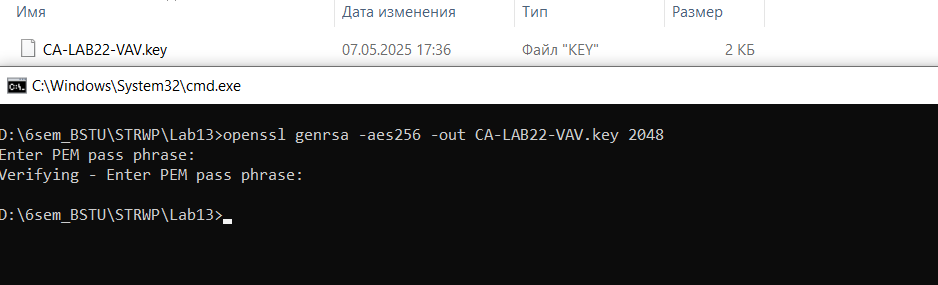
**Файл hosts:** Текстовый файл в операционной системе, который сопоставляет доменные имена с IP-адресами.

**Раунд 1: VAV – в роли CA, LPV – в роли Resource**

1. Генерация приватного ключа CA.

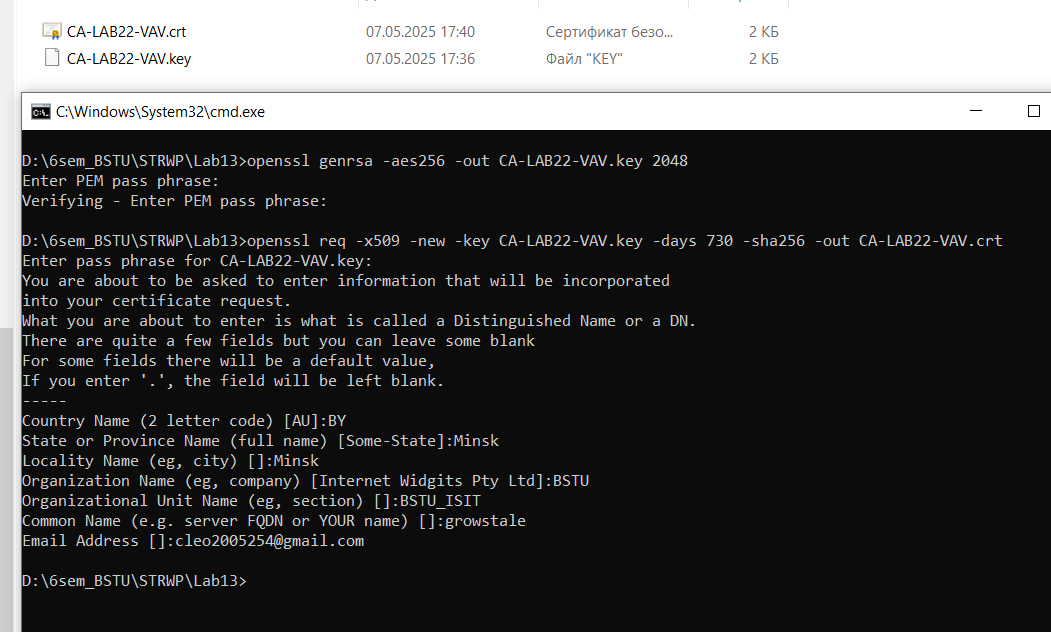
CA (Центр Сертификации):

* Приватный ключ CA: Используется для подписи сертификатов, которые CA выдает (например, сертификата для Resource). Это "печать" CA.
* Публичный ключ CA: Встроен в сертификат CA. Нужен Resource и браузеру, чтобы проверить подпись на сертификате Resource (убедиться, что он действительно выдан этим CA).



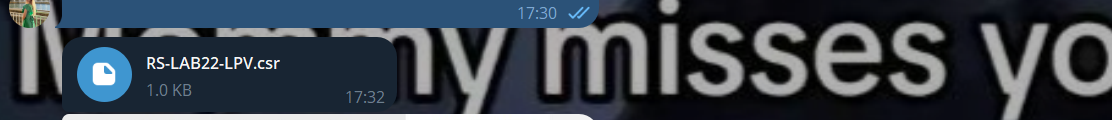
* openssl genrsa: Генерирует приватный ключ асимметричного шифрования RSA.
* -aes256: Указывает, что сгенерированный приватный ключ должен быть зашифрован с помощью симметричного алгоритма AES-256.
* -out CA-LAB22-VAV.key: Сохраняет этот зашифрованный приватный ключ в файл с именем CA-LAB22-VAV.key.
* 2048: Устанавливает длину ключа в 2048 бит (стандартная безопасная длина).

1. Генерация самоподписанного сертификата CA:



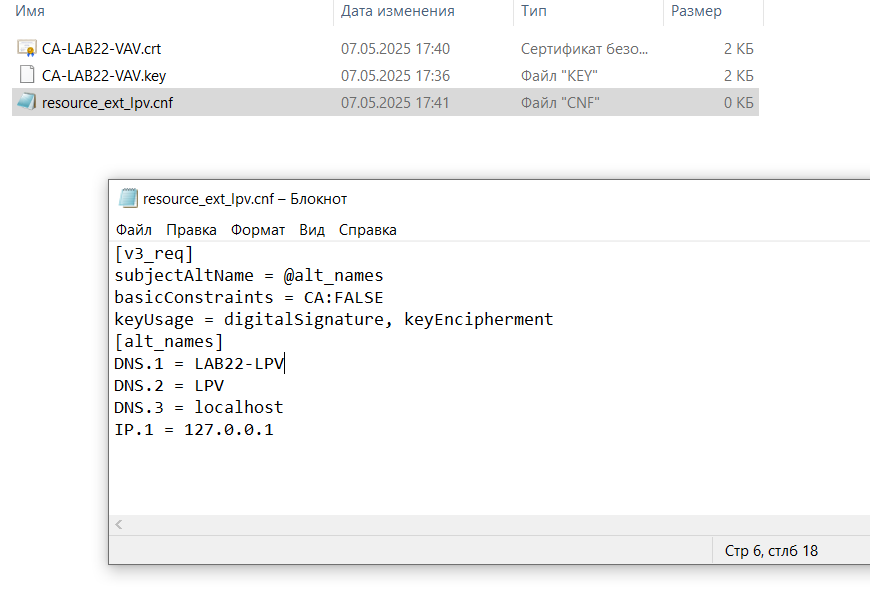
* -x509: Говорит утилите создать самоподписанный сертификат (а не запрос на сертификат). Это корневой сертификат для CA (Центра Сертификации).
* -new: Указывает, что создается новый сертификат.
* -key CA-LAB22-VAV.key: Использует ранее созданный приватный ключ CA-LAB22-VAV.key для того, чтобы подписать этот сертификат. (Так как он самоподписанный, он подписывается собственным ключом).
* -days 730: Устанавливает срок действия сертификата в 730 дней (2 года).
* -sha256: Указывает, что для создания цифровой подписи сертификата будет использоваться алгоритм хеширования SHA-256.
* -out CA-LAB22-VAV.crt: Сохраняет созданный сертификат в файл CA-LAB22-VAV.crt.

1. Получить от Resource (LPV) запрос на генерацию сертификата.

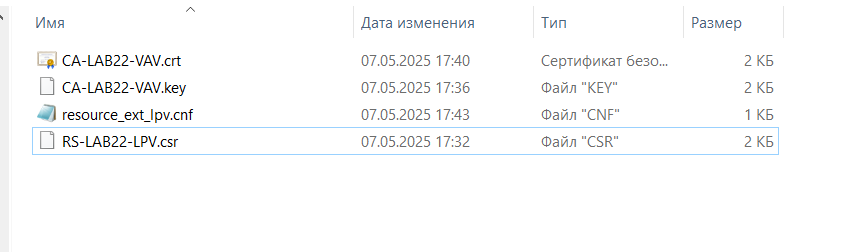


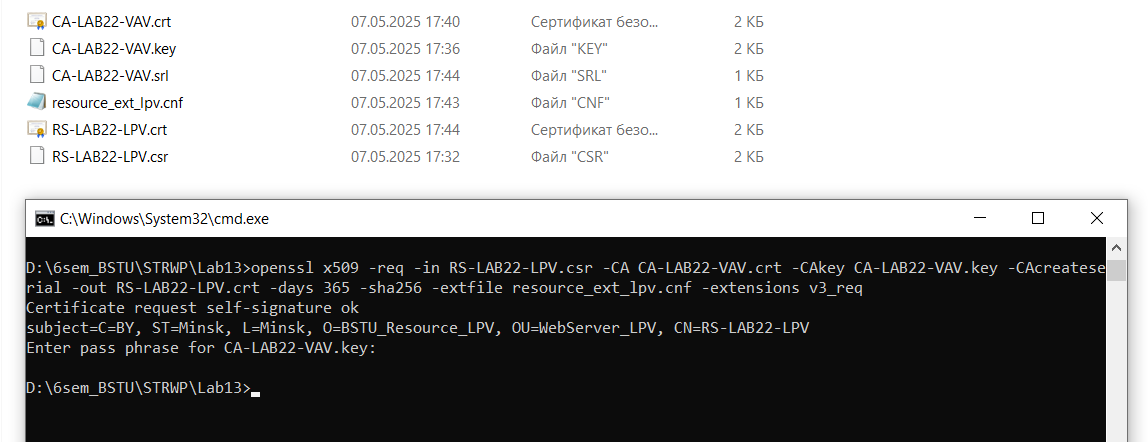
1. Сгенерировать сертификат для Resource (LPV). Разрешить доступ к доменам: LAB22-LPV и LPV.

**Конфигурационный файл для расширений сертификата**, который будем выдавать LPV.



* **subjectAltName (SAN)**: Это "Альтернативное имя субъекта". Очень важное расширение. Оно позволяет указать **несколько имен** (доменов, IP-адресов), для которых будет действителен этот сертификат.
  + **DNS.1 = LAB22-LPV**: Первое DNS-имя (домен), для которого сертификат будет действителен.
  + **DNS.2 = LPV**: Второе DNS-имя.
  + **DNS.3 = localhost**: Третье DNS-имя (полезно для локального тестирования).
  + **IP.1 = 127.0.0.1**: IP-адрес, для которого сертификат также будет действителен (тоже для локального тестирования).
* **CA:FALSE**: Критически важно! Это означает, что сертификат, который выдаётся LPV, **НЕ является сертификатом Центра Сертификации**. То есть LPV с этим сертификатом **не сможет** подписывать другие сертификаты. Это обычный сертификат для конечного пользователя/сервера.
* **keyUsage**: Определяет, для каких криптографических операций может использоваться публичный ключ, содержащийся в сертификате LPV.
  + digitalSignature: Разрешает использовать ключ для создания цифровых подписей.
  + keyEncipherment: Разрешает использовать ключ для шифрования ключей.
  + Это стандартные значения для SSL/TLS сертификата веб-сервера.





* openssl x509: Утилита для работы с сертификатами X.509.
* -req: Указывает, что на входе у нас будет запрос на сертификат (CSR).
* -in RS-LAB22-LPV.csr: Входной файл – это тот самый запрос на сертификат, который прислала LPV.
* -CA CA-LAB22-VAV.crt: Указывает сертификат моего Центра Сертификации (CA). Информация из него (например, твое имя как издателя) будет добавлена в сертификат LPV.
* -CAkey CA-LAB22-VAV.key: Указывает приватный ключ моего CA. Этим ключом будет подписываться сертификат для LPV.
* -CAcreateserial: Создает (если его нет) или использует существующий файл с серийным номером (обычно имя\_CA\_сертификата.srl). Каждый сертификат, выданный CA, должен иметь уникальный серийный номер. Эта опция автоматически управляет этим.
* -out RS-LAB22-LPV.crt: Сохраняет созданный и подписанный сертификат для LPV в файл RS-LAB22-LPV.crt.
* -days 365: Устанавливает срок действия сертификата LPV в 365 дней (1 год).
* -sha256: Использует алгоритм хеширования SHA-256 для создания цифровой подписи сертификата LPV.
* -extfile resource\_ext\_lpv.cnf: Указывает OpenSSL использовать файл конфигурации с расширениями (resource\_ext\_lpv.cnf).
* -extensions v3\_req: Говорит OpenSSL использовать секцию [v3\_req] из указанного конфигурационного файла (resource\_ext\_lpv.cnf) для добавления расширений в сертификат LPV.

1. Передать Resource (LPV) сгенерированный сертификат RS-LAB22-LPV.crt и свой CA-сертификат CA-LAB22-VAV.crt.

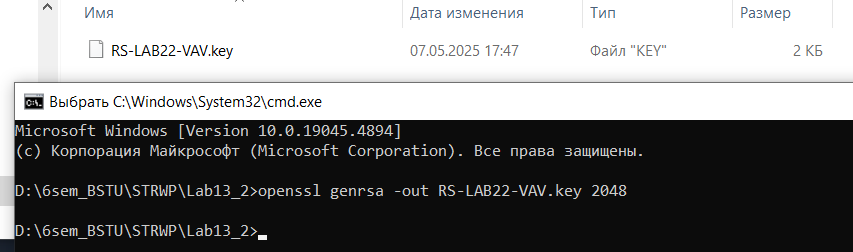


**Раунд 2: LPV – в роли CA, VAV – в роли Resource**

**Цель VAV:** Сгенерировать ключ и запрос на сертификат для своего будущего HTTPS-сервера, отправить запрос напарнице (LPV), получить от нее подписанный сертификат и сертификат ее CA, установить доверие к ее CA и запустить HTTPS-сервер.

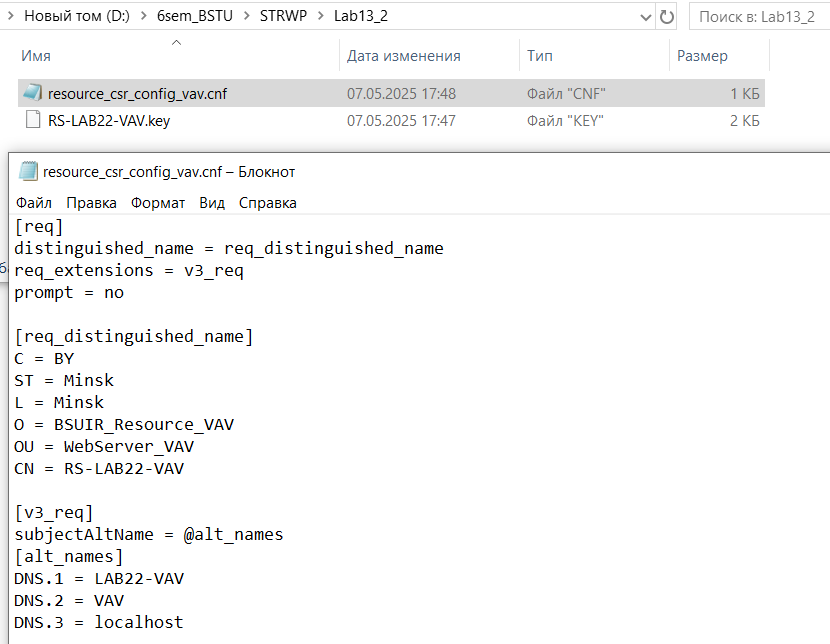
**Resource (твой веб-сервер):**

1. **Приватный ключ Resource:** Хранится на сервере. Используется для:
   * **Расшифровки** данных, которые браузер шифрует публичным ключом Resource (при установке HTTPS-соединения).
   * Для **подписи** своего запроса на сертификат (CSR), доказывая CA, что владеет этим ключом.
2. **Публичный ключ Resource:** Встроен в **сертификат Resource** (который выдал CA). Браузер использует его для:
   * **Шифрования** данных, отправляемых на сервер (чтобы только твой сервер мог их прочитать).
   * Для **проверки подлинности** сервера во время "рукопожатия" TLS.
3. Генерация приватного ключа для Resource:



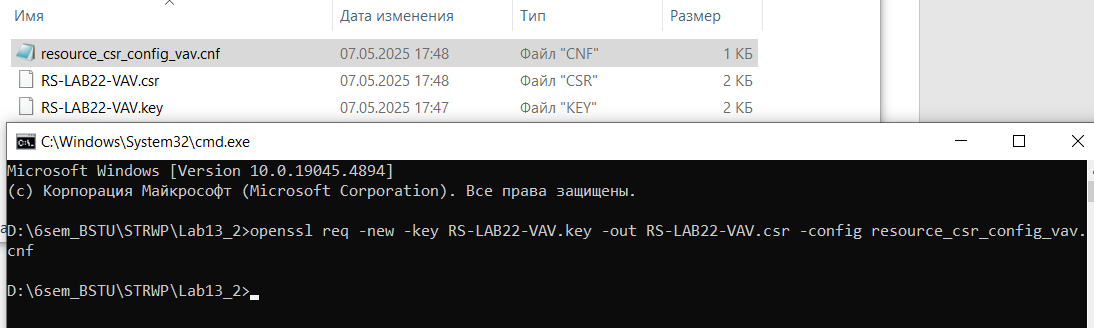
* openssl genrsa: Команда для генерации приватного ключа RSA.
* -out RS-LAB22-VAV.key: Сохраняет ключ в файл RS-LAB22-VAV.key.
* 2048: Длина ключа.
* Этот ключ будет использоваться твоим HTTPS-сервером.
* Этот ключ – секрет сервера. Соответствующий ему публичный ключ будет встроен в сертификат, который выдаст LPV.

1. Создание файла конфигурации resource\_csr\_config\_vav.cnf:



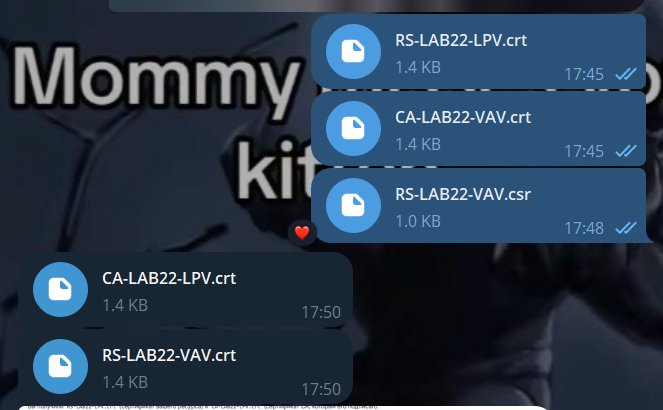
* Этот файл используется для автоматического заполнения полей CSR.
* prompt = no: Говорит openssl req не задавать вопросы интерактивно, а брать все данные из этого файла.
* [req\_distinguished\_name]: Секция с информацией о субъекте сертификата.
  + CN = RS-LAB22-VAV: Common Name для ресурса.
* [v3\_req] и [alt\_names]: Здесь я *запрашиваю* включение SAN в мой сертификат. CA (LPV) примет решение, включать их или нет, на основе своего конфигурационного файла (resource\_ext\_vav.cnf). Обычно, если запрошено и политика CA позволяет, то эти SAN будут включены.
* **Зачем:** Упрощает генерацию CSR, гарантирует, что все нужные поля заполнены правильно.

1. Генерация запроса на сертификат (CSR) для Resource (VAV):



* openssl req: Команда для создания CSR.
* -new: Создать новый CSR.
* -key RS-LAB22-VAV.key: Использовать приватный ключ RS-LAB22-VAV.key. Публичный ключ, соответствующий этому приватному, будет включен в CSR.
* -out RS-LAB22-VAV.csr: Сохранить CSR в файл RS-LAB22-VAV.csr.
* -config resource\_csr\_config\_vav.cnf: Использовать данные из файла конфигурации. Если бы его не было, openssl задавала бы вопросы про страну, город, CN и т.д.
* **Что происходит:** Утилита берет информацию из resource\_csr\_config\_vav.cnf и публичный ключ (извлекая его из RS-LAB22-VAV.key), формирует из них CSR и неявно подписывает его приватным ключом RS-LAB22-VAV.key. Это доказывает CA, что я действительно владею этим приватным ключом.

1. Передача RS-LAB22-VAV.csr CA (LPV).

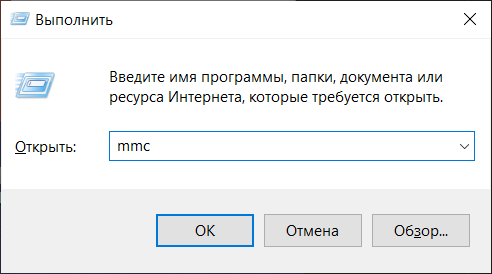


1. Получить от CA (LPV) два сертификата.

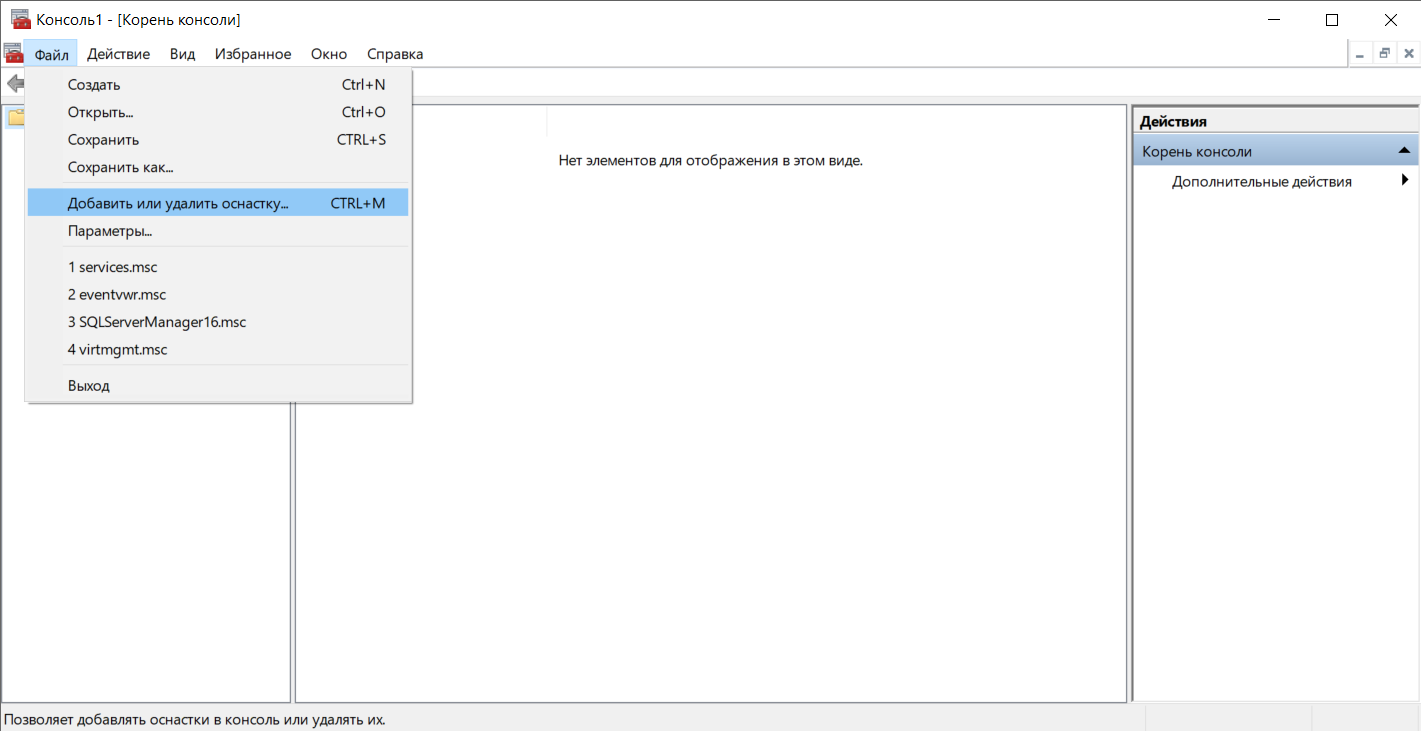
* RS-LAB22-VAV.crt (мой подписанный сертификат сервера)
* CA-LAB22-LPV.crt (корневой сертификат CA LPV)

1. Импортировать CA-сертификат (CA-LAB22-LPV.crt) в хранилище сертификатов (доверенные центры сертификации) на компьютере.

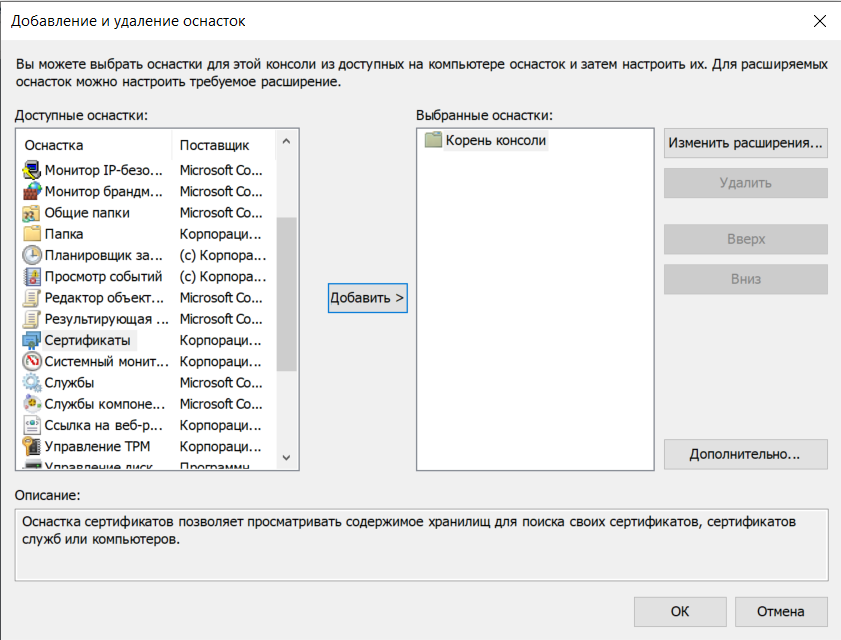
* Нажать Win+R, ввести mmc и нажать Enter.



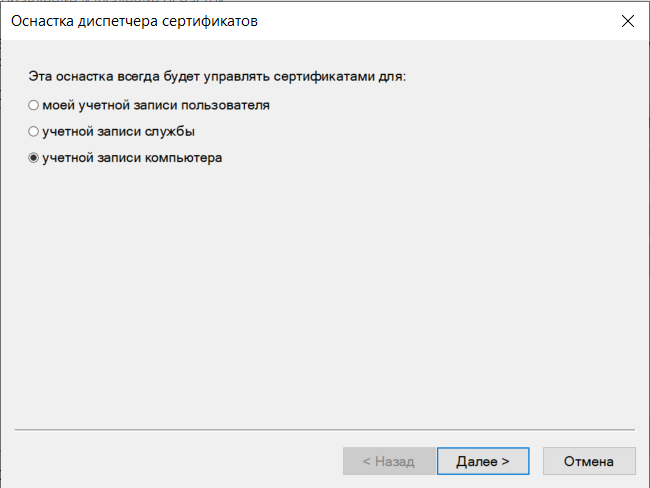
* В консоли: Файл -> Добавить или удалить оснастку....

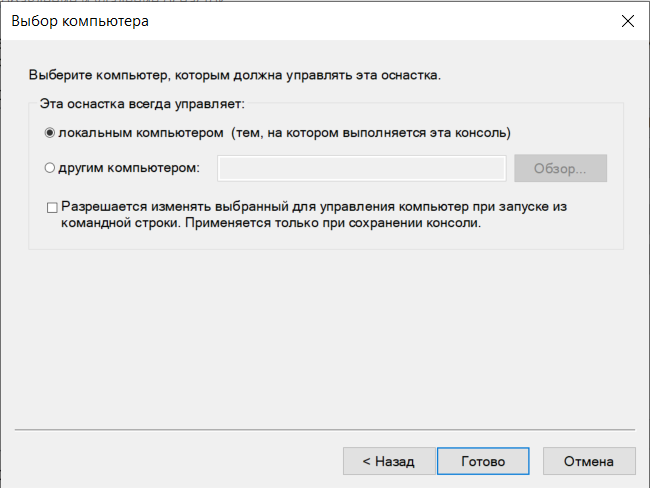


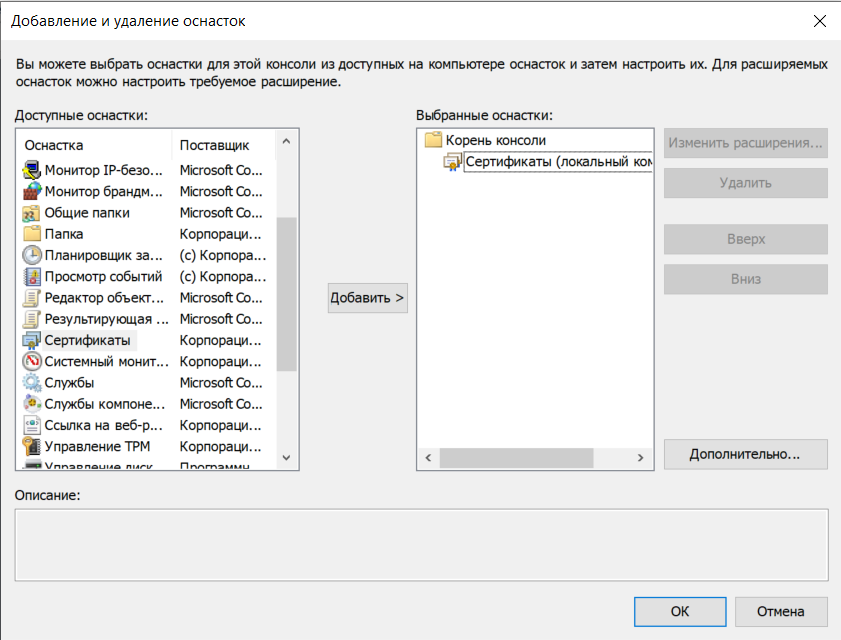
* Выбрать Сертификаты, нажать Добавить >.



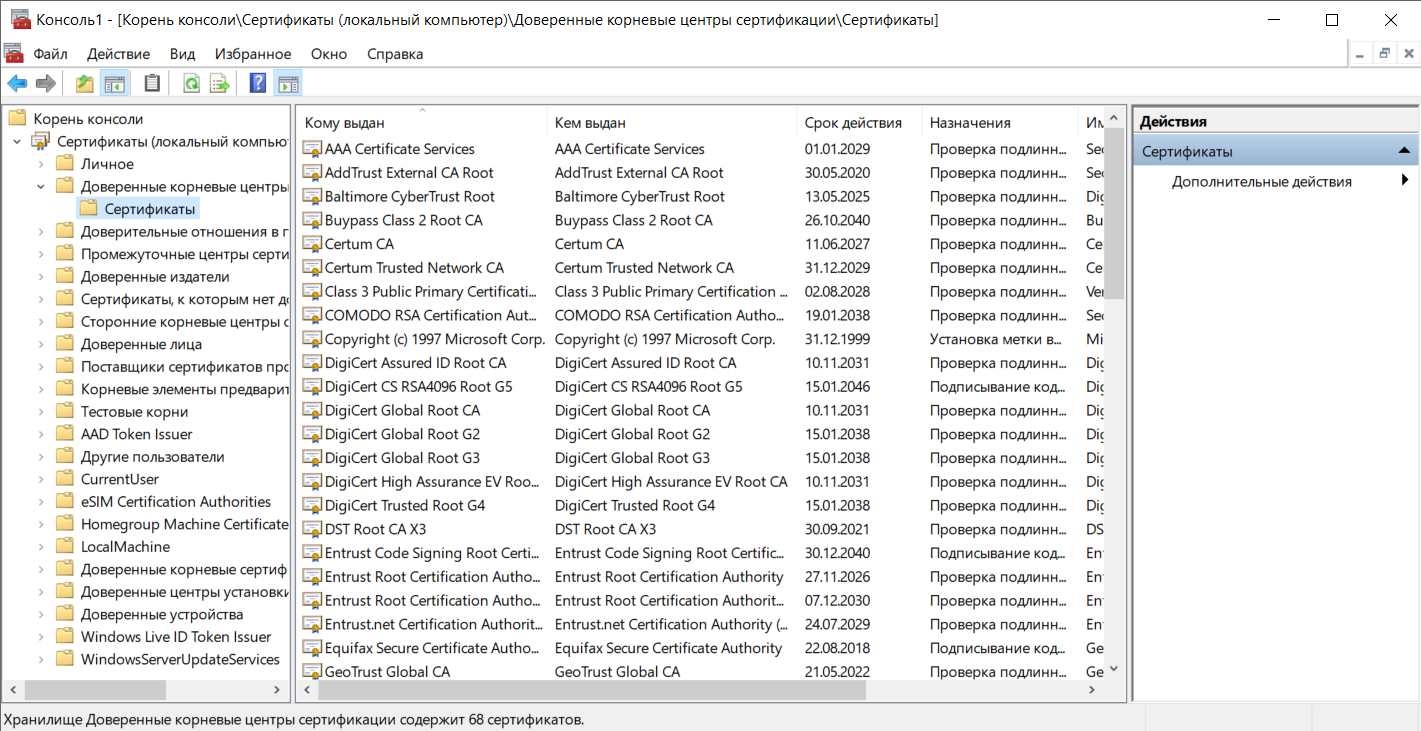
* Выбрать учетной записи компьютера, Далее, Готово, ОК.



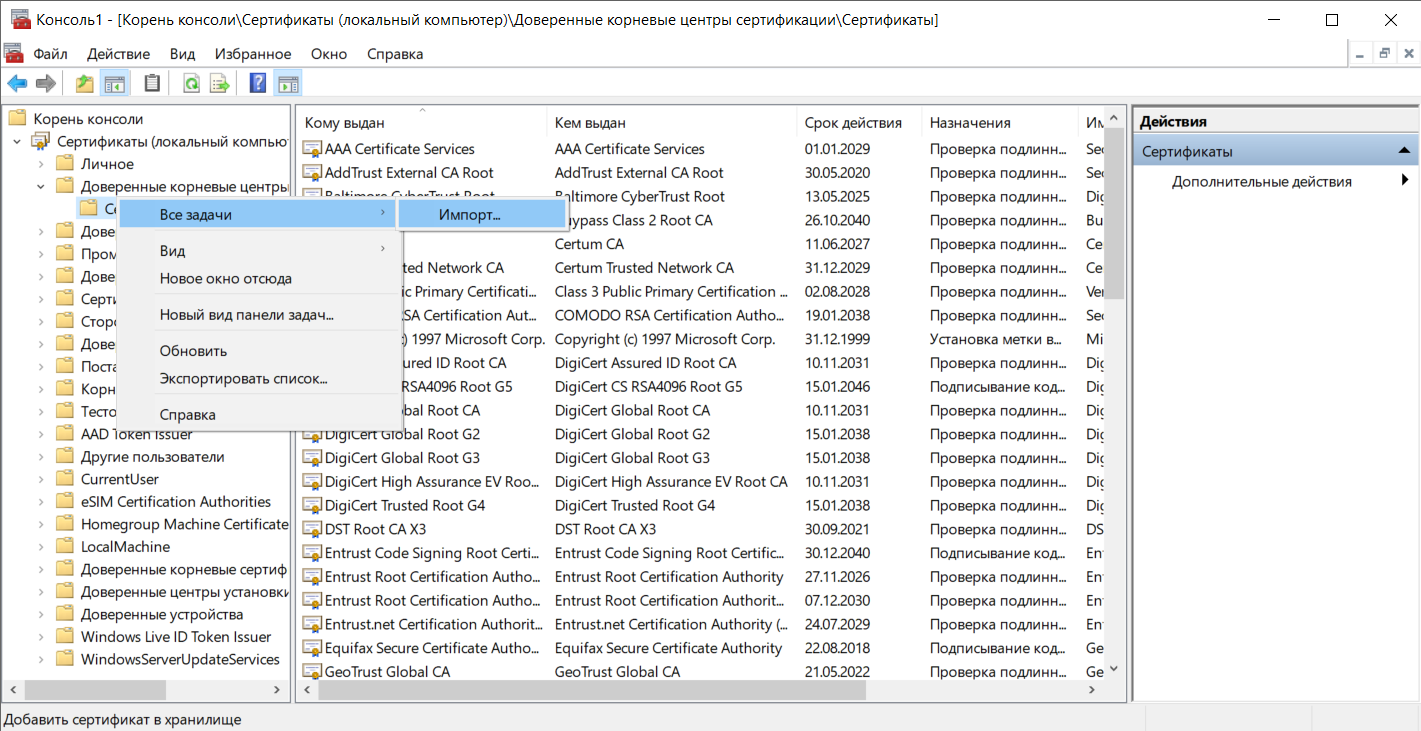




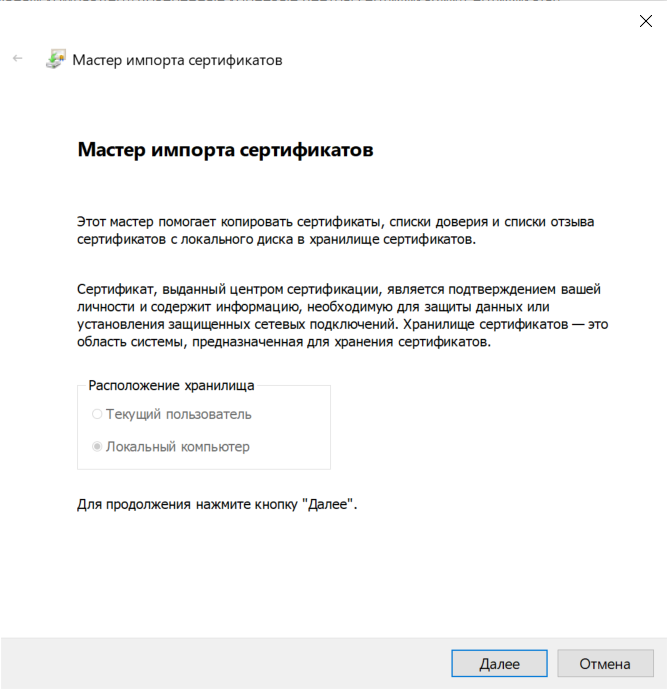
* В дереве слева: Сертификаты (локальный компьютер) -> Доверенные корневые центры сертификации -> Сертификаты.

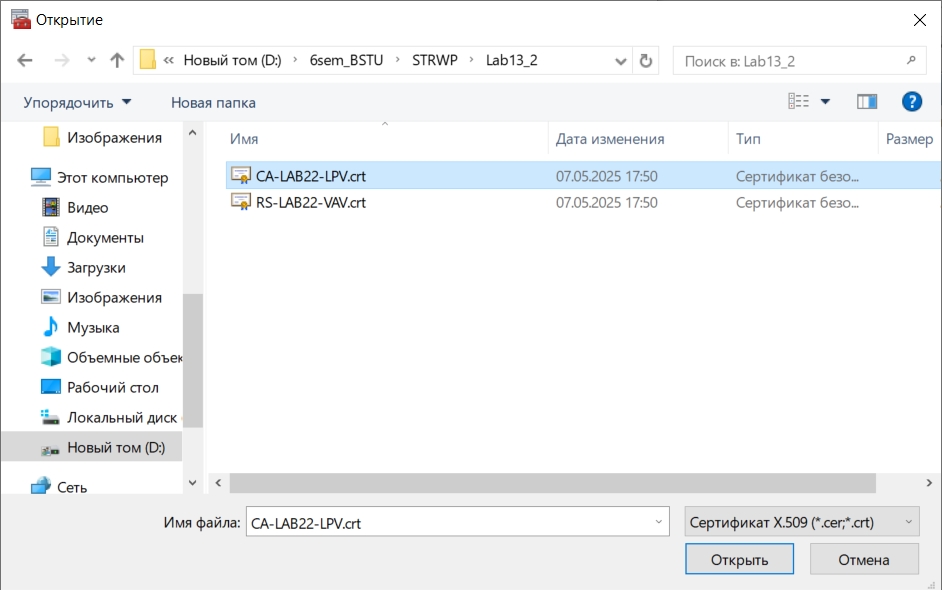


* Правой кнопкой мыши на Сертификаты -> Все задачи -> Импорт....

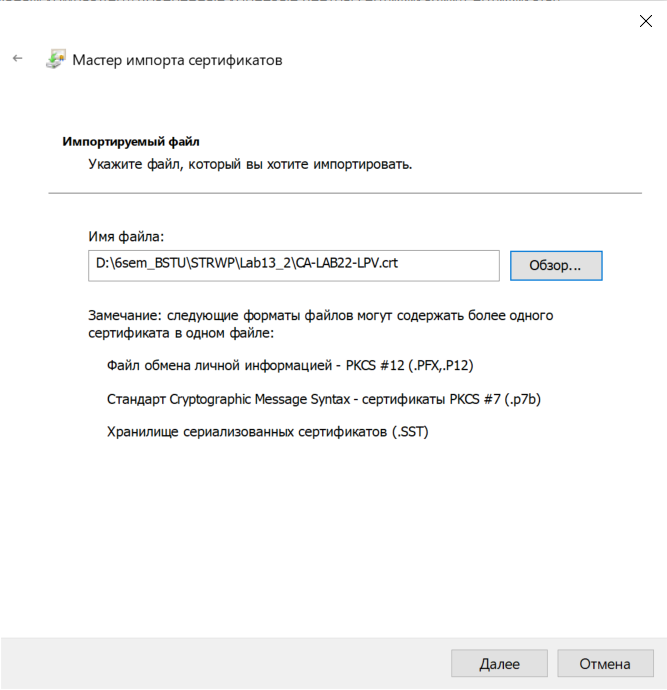


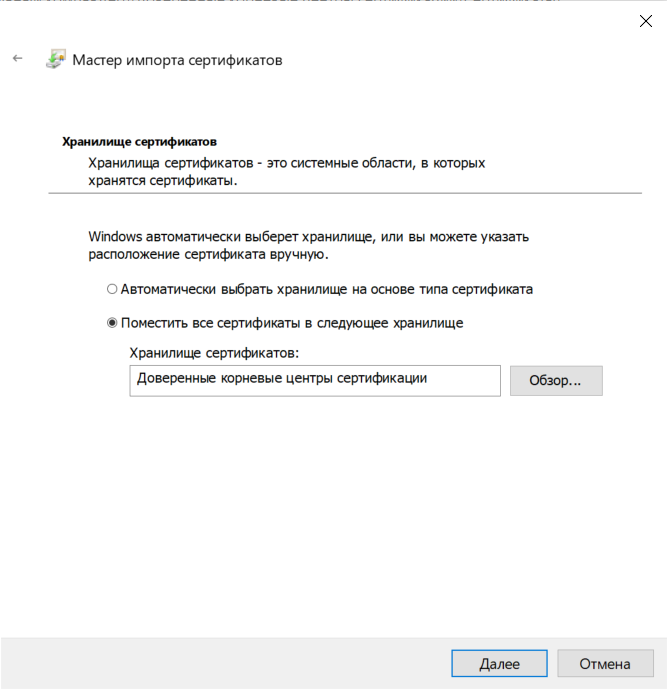
* Мастер импорта: Далее -> Обзор... (выбрать файл CA-LAB22-LPV.crt) -> Далее.

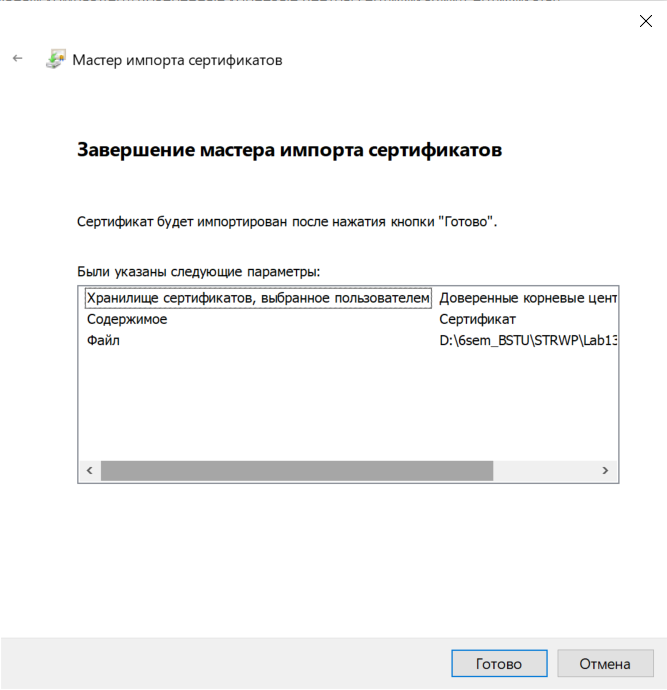




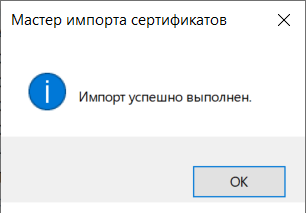
* Убедиться, что выбрано хранилище "Доверенные корневые центры сертификации".





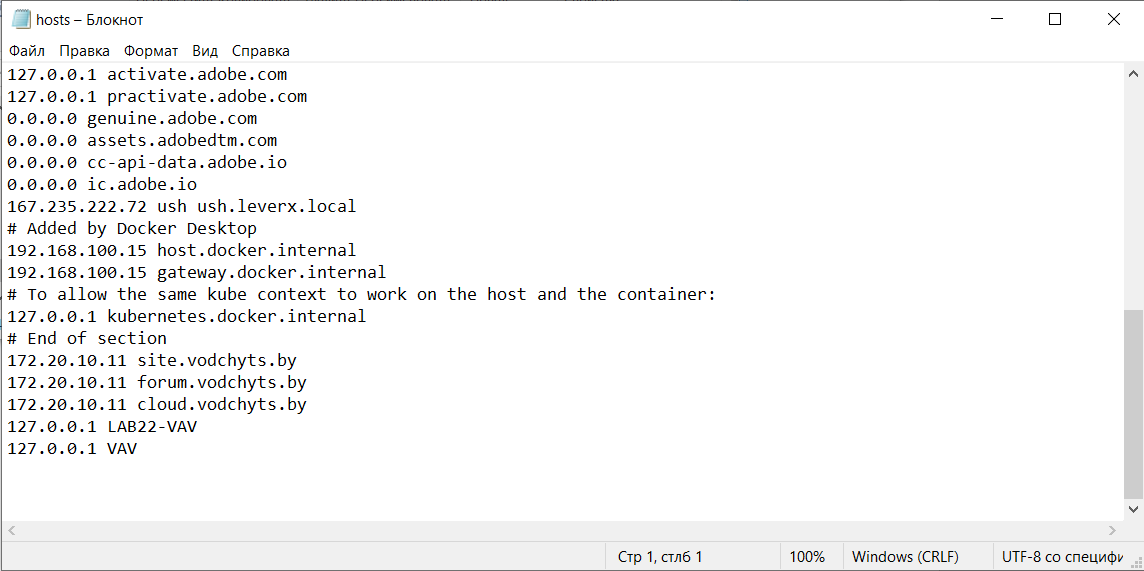


* Далее -> Готово. Согласиться с предупреждением безопасности (если будет).



* Что происходит: Я сообщаю своей операционной системе, что CA с сертификатом CA-LAB22-LPV.crt является доверенным.
* Зачем: Теперь, когда браузер (или другая программа) увидит сертификат (RS-LAB22-VAV.crt), подписанный этим CA (CA-LAB22-LPV), он проверит цепочку доверия. Он увидит, что RS-LAB22-VAV.crt подписан CA-LAB22-LPV. Затем он посмотрит в свои доверенные корневые центры и найдет там CA-LAB22-LPV.crt. Цепочка замкнется, и сертификат будет считаться доверенным. Без этого шага браузер выдал бы ошибку "Недоверенный сертификат" или "NET::ERR\_CERT\_AUTHORITY\_INVALID".

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



* **Что происходит:** Я говорю своему компьютеру, что доменные имена LAB22-VAV и VAV соответствуют IP адресу 127.0.0.1.
* **Зачем:**  HTTPS-сервер будет работать на моём компьютере. Чтобы браузер мог найти его по именам LAB22-VAV и VAV (которые указаны в SAN сертификата), нужна эта запись. Иначе браузер попытается найти эти домены в интернете через DNS и не найдет.

