Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение   
высшего профессионального образования   
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»  
Институт радиоэлектроники и информационных технологий - РТФ

Лабораторная работа  
«Удостоверяющий центр (Certificate authority). OpenSSL.»  
по дисциплине «Криптографические методы защиты информации»

Екатеринбург 2020

# Теоретические сведения к лабораторной работе

Удостоверяющий центр

**Центры сертификации** (или удостоверяющие центры, CA) выдают цифровые сертификаты, отвечают за управление криптографическими ключами пользователей.

**Цифровые сертификаты** — это верифицируемые файлы данных, которые содержат сведения, чтобы помочь веб-сайтам, людям и устройствам подтвердить свою идентификацию, упрощают задачу определения принадлежности ключей предполагаемым владельцам.

Использование удостоверяющего центра:



Здесь PKI (Public Key Infrastructure) — инфраструктура открытого ключа (ИОК), набор средств, распределённых служб и компонентов, в совокупности используемых для поддержки криптозадач на основе закрытого и открытого ключей.

(Открытый (публичный ключ) доступен всем. Используется для шифрования данных при обращении браузера к серверу.

Закрытый (секретный ключ) известен только владельцу сайта. Используется для расшифровки данных, отправленных браузером.)

Издание УЦ SSL- сертификатов

Для обеспечения безопасности передачи данных между клиентом и сервером поверх открытых сетей существует технология SSL (Secure Sockets Layer) или TLS (Transport Layer Security)

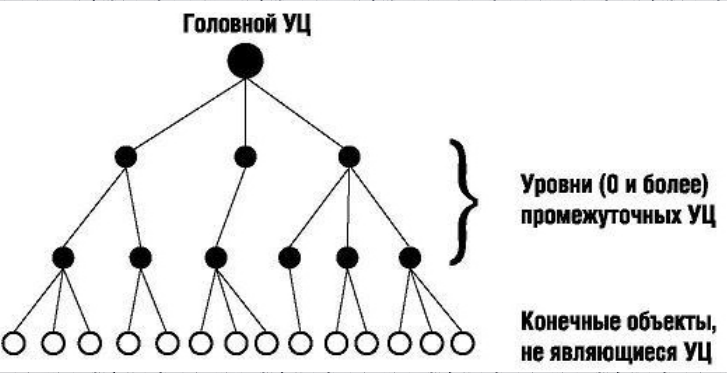
У центра сертификации должен быть свой электронный документ, которым он будет подписывать SSL- сертификаты при их выдаче. Этим документом является **корневой сертификат** (в ЛР это будет самоподписанный сертификат на приватный ключ шифрования RSA)

SSL-сертификаты выпускают доверенные центры сертификации, но каждый может сгенерировать бесплатный SSL-сертификат самостоятельно, но, если корневой сертификат центра не установлен в браузере — он будет распознаваться как недостоверный.

(В данной лабораторной работе ключи RSA и запросы на создание сертификатов (CSR-запросы) будут создаваться с помощью набора инструментов командной строки криптографической библиотеки OpenSSL, которая является open source реализацией двух протоколов SSL и TLS.)

Промежуточный УЦ

В части случаев конечный сертификат выдает не основной центр, а посредник. Создается цепочка доверия: корневым сертификатом подписывается промежуточный, промежуточным — сертификат, например, сайта.



Посредник в данном случае – это тоже удостоверяющий центр, только **промежуточный** и для проверки его сертификата создается цепной сертификат, который доказывает его связь с корневым удостоверяющим центром.

Веб-серверы. Сертификаты веб-сервера.

**Сервер криптографических ключей** — это хост, предназначенный для хранения и передачи пользователям, а также другим криптографическим серверам криптографических ключей.

**Ключи** (называемые «ключи веб-сервера»), распространяемые данным видом сервера, чаще всего используются как часть цифрового сертификата, содержащего не только сам ключ, но также и информацию о владельце ключа. Как правило, в этом случае используется сертификат одного из распространённых стандартов: OpenPGP, X.509 или PKCS. Кроме того, обычно данные ключи являются открытыми ключами для использования в алгоритмах шифрования систем с открытым ключом.

Чтобы получить сертификат, необходимо сформировать запрос.

**Запрос на сертификат** - сообщение, содержащее необходимую информацию для получения сертификата в Удостоверяющем центре.

При формировании запроса необходимо указать, по какому стандарту должен быть сформирован сертификат. В нашем случае будет использоваться один из самых популярных стандартов – X.509.

**X.509** — стандарт для инфраструктуры открытого ключа и инфраструктуры управления привилегиями.

X.509 определяет стандартные форматы данных и процедуры распределения открытых ключей с помощью соответствующих сертификатов с цифровыми подписями. Кроме того, X.509 определяет формат списка аннулированных сертификатов, формат сертификатов атрибутов и алгоритм проверки подписи путём построения пути сертификации. X.509 предполагает наличие иерархической системы удостоверяющих центров для выдачи сертификатов.

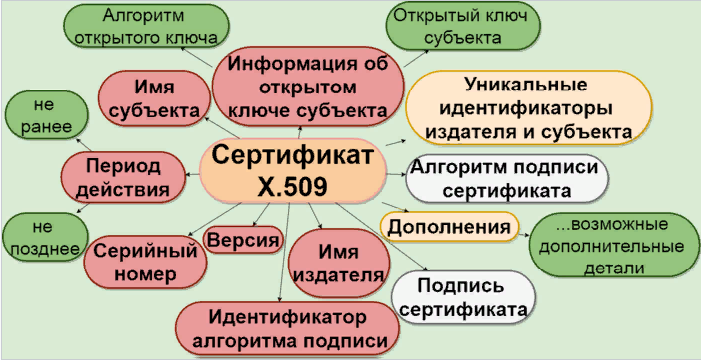


Рисунок 1 – общий стандарт для интернета, использующий сертификат x509

Внедрение сертификата и его отзыв.

**SSL-сертификаты** применяются для обеспечения безопасного соединения и передачи данных в зашифрованном виде. Чаще всего они устанавливаются на веб- и почтовые сервера для обеспечения возможности работы сайта по защищенному протоколу https. Наиболее распространённой областью применения SSL-сертификатов является защита HTTP-трафика, для чего сертификат устанавливается на Web-сервер и используется для шифрования трафика между сервером и клиентом. Корпоративная почта и системы групповой работы, системы хранения документов и всевозможные Web-приложения - все они нуждаются в защите HTTP-трафика.

Поэтому защита внутренних ресурсов обычно начинается с внедрения корпоративных SSL-сертификатов.

Как и в случае с любыми другими удостоверяющими документами, кроме возможности выдать сертификат, должна быть возможность прекратить его действие. Прекращение действия сертификата называется его **отзывом,** и в каждый момент времени существует файл, называемый списком отзыва (CRL), содержащий перечень сертификатов, действие которых прекращено. Ссылка на этот файл может быть вставлена непосредственно в сертификат (параметр crlDistributionPoint В секции ssl\_server и ssl\_client), кроме того, многим программам необходимо задавать ссылку на список отзыва в явном виде.

# Практическое задание

Данная работа выполняется на виртуальной машине, на которую необходимо предварительно установить операционную систему Linux Ubuntu и библиотеку OpenSSL.

1. Создание удостоверяющего центра (УЦ, certificate authority)

*1.1. Создание файловой иерархии*

1.1.1. Для удостоверяющего центра необходима директория, в которой будут храниться связанные с ним файлы. У нас такой директорией будет являться /etc/pki/CA. Чтобы её создать, введём в командной строке следующее:

su

mkdir /etc/pki /etc/pki/CA

1.1.2. Создадим иерархию в нашей файловой системе. Храниться данные будут, главным образом, в подкаталогах certs — для сертификатов, crl — для списка отзывов сертификатов, newcerts — для поступивших запросов, private — для личного ключа (требует особой защиты). Для этого пользуемся следующими командами:

cd /etc/pki/CA

mkdir certs crl newcerts private

1.1.3. Устанавливаем права доступа к подкаталогу Private следующей командой:

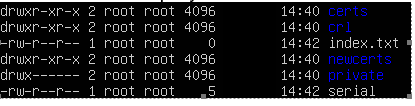
chmod 700 private

1.1.4 Также создадим БД для сертификатов в файле index.txt и файл серийных номеров сертификатов serial:

touch index.txt

echo 1000 > serial

В результате выполнения пунктов должен получиться следующий результат:



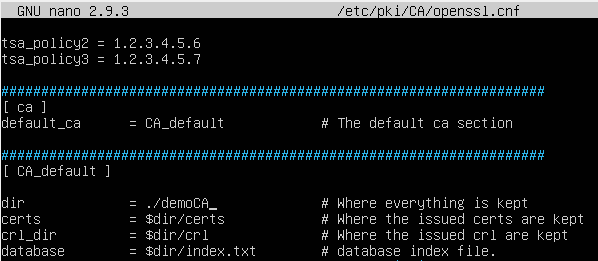
1.1.5 Также нам понадобится конфигурационный файл openssl.cnf. Найдём его при помощи команды

openssl ca

1.1.6 Скопируем данный файл в только что созданный каталог CA и исправим строку dir=./… в секции [ CA\_default ] на

dir=/etc/pki/CA

Убедитесь, что другие имена каталогов в этой секции соответствуют созданной вами иерархии.





*1.2.* *Корневой сертификат*

*Начинаем самое сложное.*

1.2.1 На этом этапе мы будем генерировать личный ключ и он требует особой защиты. В противном случае, атакующий получит к нему доступ и сможет издавать сертификаты. Генерируем ключ и защищаем его шифром AES-256:

openssl genrsa -aes256 -out /etc/pki/CA/private/ca.key.pem 4096

1.2.2. Разрешаем только чтение для владельца данного ключа установкой прав 400

chmod 400 /etc/pki/CA/private/ca.key.pem

1.2.3 Издаём корневой самоподписанный сертификат, который будет действовать 10 лет, с расширением v3\_ca. Расширение указывает на то что, это - сертификат УЦ.

По требованию системы введите свои данные.

openssl req -new -x509 -days 3650 -key /etc/pki/CA/private/ca.key.pem -extensions v3\_ca -out /etc/pki/CA/ca.cert.pem

1.2.4 Устанавливаем права на чтение данного сертификата для владельца, пользователя и групп пользователей установкой прав 444

chmod 444 ca.cert.pem

2. Промежуточный УЦ

2.1. Создание файловой иерархии

Для создания промежуточного удостоверяющего центра используем тот же метод, что и в пункте 1.1, с тем отличием, что файлы промежуточного УЦ будут храниться в директории Intermediate.

2.1.1 Создаём директорию Intermediate и иерархию в нашей файловой системе. Напомним значение подкаталогов: certs — для сертификатов, crl — для списка отзывов сертификатов, newcerts — для поступивших запросов, private — для личного ключа (требует особой защиты). Для этого пользуемся следующими командами:

mkdir /etc/pki/CA/intermediate

cd /etc/pki/CA/intermediate

mkdir certs crl newcerts private

2.1.2. Устанавливаем права доступа к подкаталогу Private следующей командой:

chmod 700 private

2.1.3 Также создадим БД для сертификатов в файле index.txt и файл серийных номеров сертификатов serial:

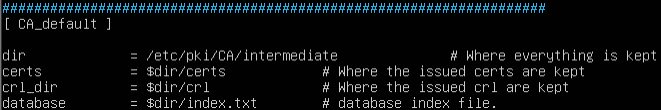
touch index.txt

echo 1000 > serial

2.1.4 Копируем файл openssl.cnf в директорию Intermediate

cp ../openssl.cnf .

Не забудьте исправить в данном файле секции по примеру, представленному в пункте 1.1.6



*2.2. Сертификат промежуточного УЦ*

2.2.1 Создаём сертификат промежуточного УЦ по аналогии с пунктом 1.2

cd /etc/pki/CA/intermediate

openssl genrsa -aes256 -out private/ca.key.pem 4096

chmod 400 private/ca.key.pem

openssl req -new -key private/ca.key.pem -out intermediate.csr.pem

2.3. Выдача сертификата корневым УЦ

cd /etc/pki/CA

2.3.1 Издаем сертификат с расширением v3\_ca (указывает на то что, это - сертификат УЦ), используя созданные ранее ключ, корневой сертификат и сертификат промежуточного УЦ. Используем формат криптографического хэширования SHA1. Для этого воспользуемся следующей командой:

openssl ca -config openssl.cnf \

-keyfile private/ca.key.pem -cert ca.cert.pem \

-extensions v3\_ca -notext -md sha1 \

-in intermediate/intermediate.csr.pem \

-out intermediate/ca.cert.pem

2.3.2 Устанавливаем права на чтение данного сертификата для владельца, пользователя и групп пользователей установкой прав 444

chmod 444 intermediate/ca.cert.pem

2.3.3 Проверим действительность сертификата следующей командой:

openssl verify -CAfile /etc/pki/CA/ca.cert.pem /etc/pki/CA/intermediate/ca.cert.pem



2.4. Цепной сертификат

2.4.1 При проверке сертификата промежуточного УЦ может потребоваться цепной сертификат в случае, если у клиента есть доверие только корневому УЦ. Создадим такой сертификат самостоятельно и установим к нему права доступа:

cd /etc/pki/CA

cat intermediate/cacert.pem ca.cert.pem > intermediate/ca-chain.cert.pem

chmod 444 intermediate/ca-chain.cert.pem

2.4.2 Проверим так же действительность цепного сертификата

следующей командой:

openssl verify -CAfile /etc/pki/CA/ca.cert.pem /etc/pki/CA/intermediate/ca-chain.cert.pem

3. Создание и подписывание сертификатов промежуточным УЦ

3.1. Создание личного ключа веб-сервера

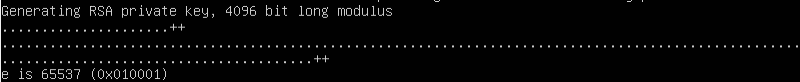
3.1.1 Генерируем ключ веб-сервера localhost.key.pem, прибегнув к уже использованной ранее команде genrsa.

cd /etc/ssl/localcerts

openssl genrsa -out localhost.key.pem 4096

3.1.2 Устанавливаем права доступа 400

chmod 400 localhost.key.pem



*3.2. Создание запроса на сертификат*

3.2.1 Создадим запрос на сертификат, используя команду req:

openssl req -new -key localhost.key.pem -out localhost.csr.pem

*3.3. Издание сертификата промежуточным УЦ*

3.3.1 Теперь, чтобы ответить на запрос, издаём сертификат от имени промежуточного УЦ. Для этого с помощью команды ca создадим файл сертификата.

cd /etc/pki/CA/intermediate

openssl ca -config openssl.cnf -notext -md sha1 \

-in /etc/ssl/localcerts/localhost.csr.pem -out /etc/ssl/localcerts/localhost.cert.pem

3.3.2 Установим ограниченные права доступа на сертификат.

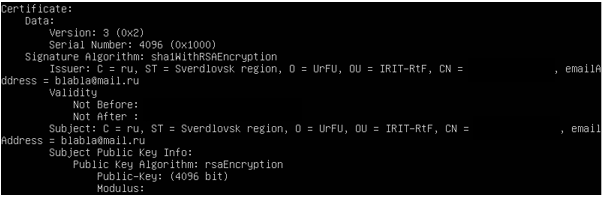
chmod 444 /etc/ssl/localcerts/localhost.cert.pem

*3.4. Сертификат. Просмотр. Проверка.*

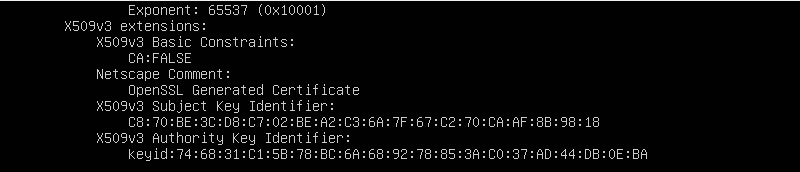
3.4.1 Чтобы убедиться в том, что сертификат заполнен верно, необходимо его проверить. Для этого воспользуемся некоторыми командами для его просмотра и визуальной проверки:

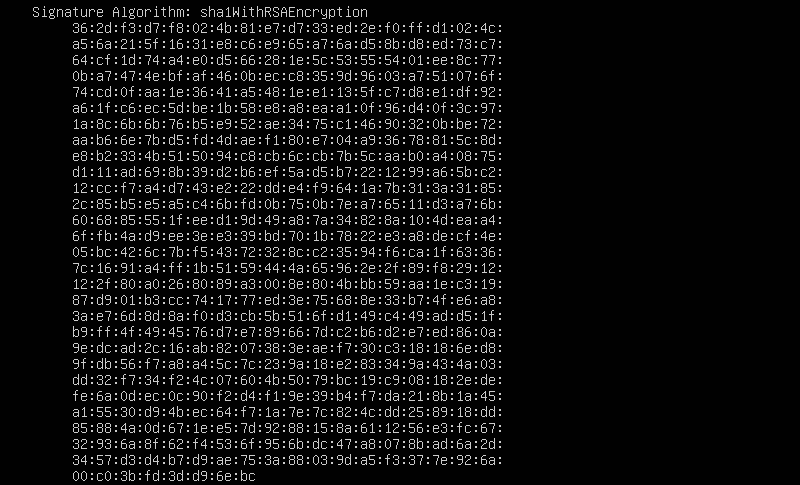
cd /etc/ssl/localcerts

openssl x509 -in localhost.cert.pem -noout -text | more



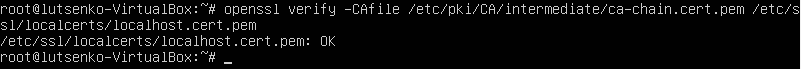






3.4.2 Проверяем сертификат уже знакомой нам командой Verify

openssl verify -CAfile /etc/pki/CA/intermediate/ca-chain.cert.pem localhost.cert.pem



*3.5. Добавление сертификата в конфигурацию веб-сервера*

3.5.1 Отредактируем конфигурационный файл из директории /etc/apache2/sites-enabled/default-ssl.conf, чтобы привести его к следующему виду:

…

<VirtualHost \_default\_:443>

ServerAdmin webmaster@localhost

ServerName localhost:443

DocumentRoot /var/www/html

…

SSLEngine On

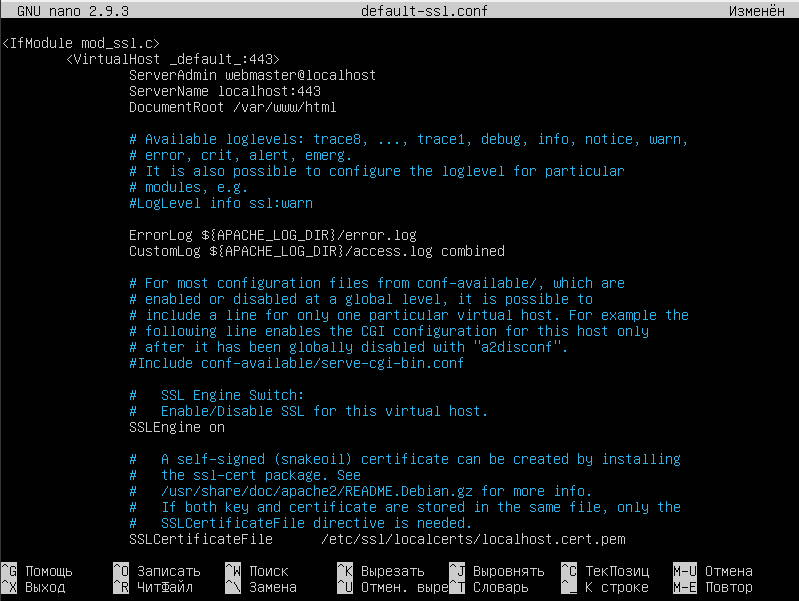
…

SSLCertificateFile /etc/ssl/localcerts/localhost.cert.pem

SSLCertificateKeyFile /etc/ssl/localcerts/localhost.key.pem

SSLCACertificateFile /etc/pki/CA/intermediate/ca.cert.pem

…



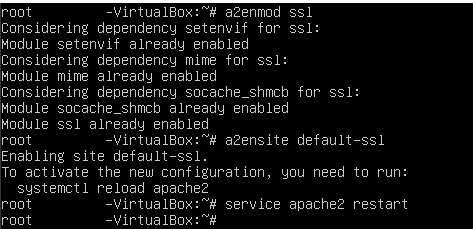


3.5.2 Запускаем веб-сервер с включённой поддержкой защищённых соединений:

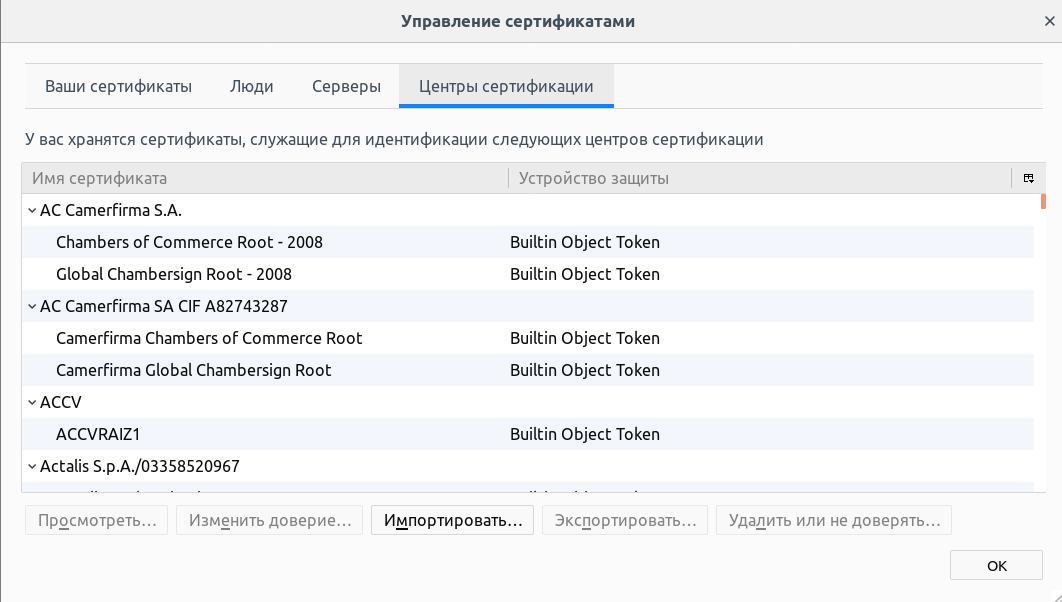
sudo a2enmod ssl

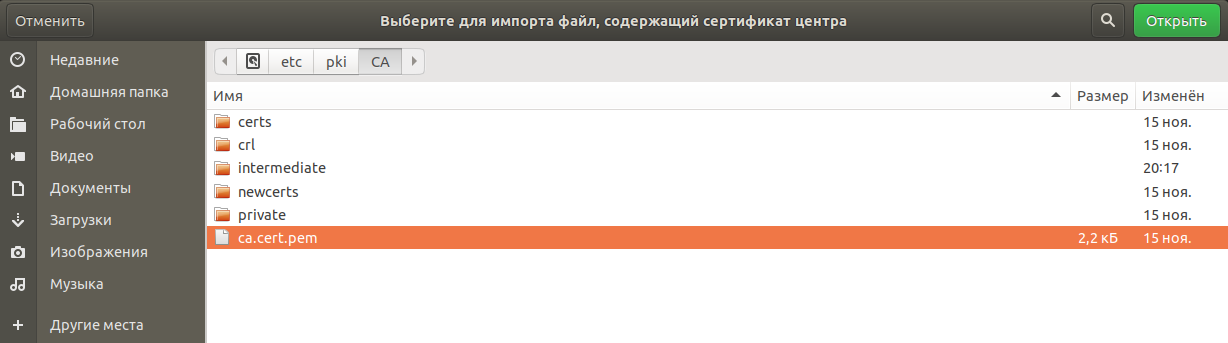
sudo a2ensite default-ssl

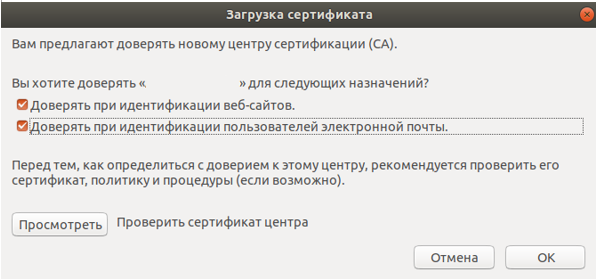
sudo service apache2 restart

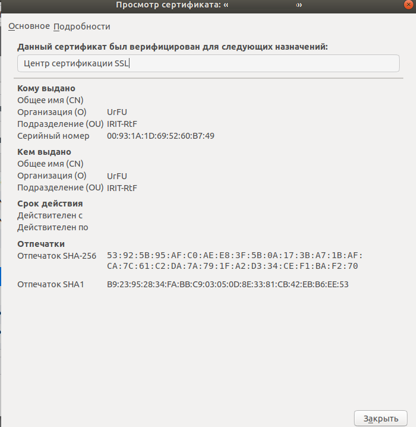


3.5.3 Нам необходимо проверить защищённое соединение. Для этого воспользуемся браузером, в который предварительно импортируем сертификат корневого УЦ.









4. Отзыв сертификатов

*4.1. Список отозванных сертификатов*

4.1.1 При процедуре отзыва сертификата обязательно должен присутствовать список сертификатов, которые будут отозваны. Создадим его при помощи функции ca, конкретно – при помощи атрибута gencrl, генерирующего список отозванных сертификатов.

cd /etc/pki/CA

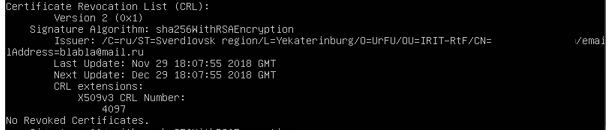
openssl ca -keyfile private/ca.key.pem -cert certs/ca.cert.pem \

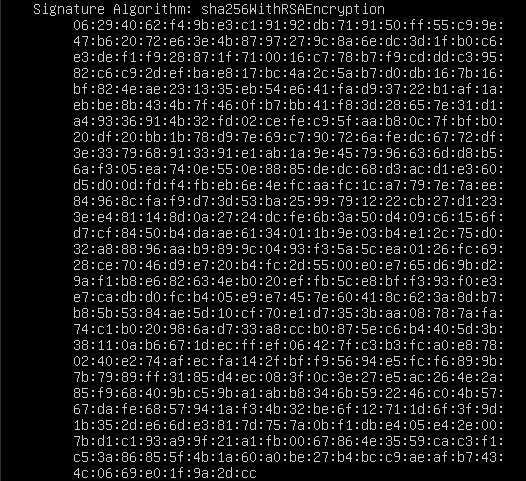
-gencrl -out crl/crl.pem

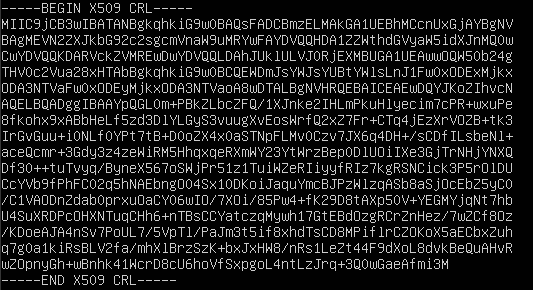


4.1.2 Для просмотра получившегося списка введите следующую команду:

openssl crl -in crl/crl.pem -text





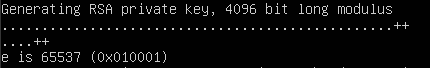


*4.2. Создание сертификата пользователя*

4.2.1 Создадим ключ пользователя Alice и установим к нему ограниченные права доступа.

openssl genrsa -out private/alice.key.pem 4096

chmod 400 private/alice.key.pem

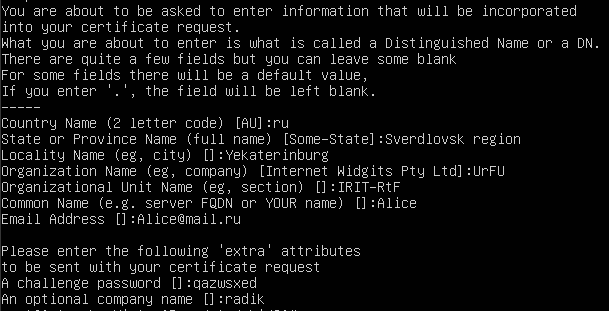


4.2.2 Создадим также запрос на сертификат от пользователя Alice с помощью команды req.

openssl req -new \

-key private/alice.key.pem \

-out certs/alice.csr.pem

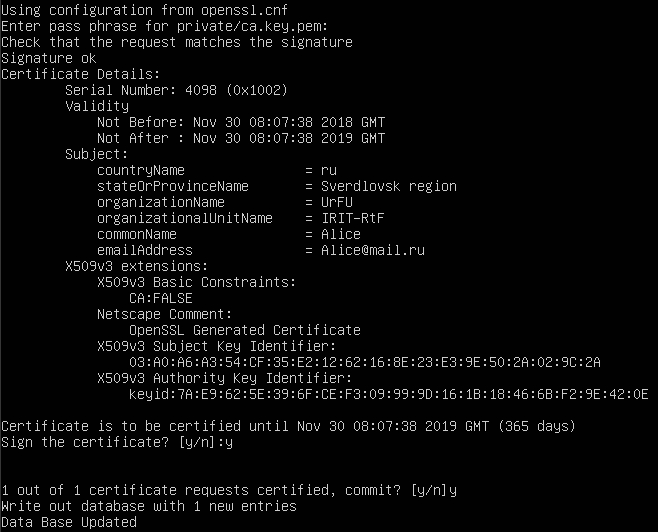


4.2.3 Выдаём сертификат по ранее сформированному запросу.

openssl ca -config openssl.cnf -keyfile private/ca.key.pem -cert ca.cert.pem \

-extensions usr\_cert -notext -md sha1 \

-in certs/alice.csr.pem -out certs/alice.cert.pem



4.2.4 Ограничим права доступа к сертификату.

chmod 444 certs/alice.cert.pem

4.2.5 Теперь проверьте базу данных сертификатов. Появился ли там выданный сертификат?

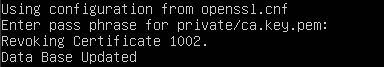
cat index.txt

*4.3. Отзыв сертификата пользователя*

4.3.1 Жизнь жестока, и порой сертификат требуется отозвать. Проделаем эту процедуру с сертификатом, который только что выдали на пользователя Alice.

openssl ca -config openssl.cnf -keyfile private/ca.key.pem -cert ca.cert.pem \

-revoke certs/alice.cert.pem



4.3.2 Что изменилось в записи о пользователе с выполнением данной команды? Поясните.

cat index.txt

*4.4. Обновление списка отозванных сертификатов*

4.4.1 После отзыва сертификата мы обязательно должны указать в своей базе, что он был отозван. Для этого обновим список отозванных сертификатов следующей командой:

openssl ca -config openssl.cnf -keyfile private/ca.key.pem -cert ca.cert.pem \

-gencrl -out crl/crl.pem



4.4.2 Просмотрите список. Что в нём изменилось?

openssl crl -in crl/crl.pem -text

Всем спасибо, подписывайтесь, ставьте лайки, делайте красную кнопочку серой и нажимайте на колокольчик