Создание сервера генерации цифровых сертификатов X.509 с использованием библиотеки SSL

Выполнил студент группы ИКТ-42

Елецких Евгений

Что такое X.509

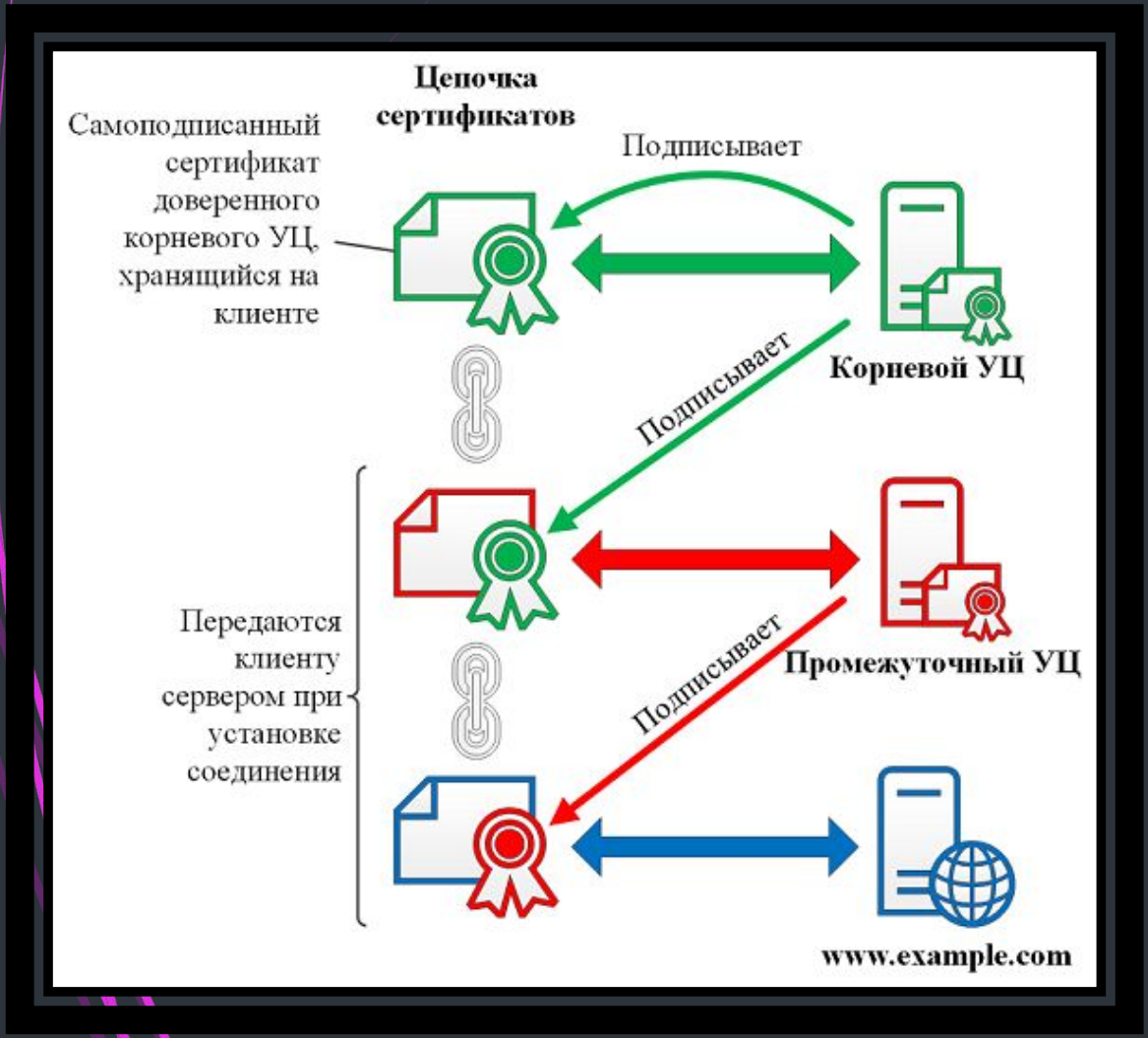
- X.509 — стандарт ITU-T для инфраструктуры открытого ключа (англ. Public key infrastructure, PKI) и инфраструктуры управления привилегиями (англ. Privilege Management Infrastructure).
- X.509 определяет стандартные форматы данных и процедуры распределения открытых ключей с помощью соответствующих сертификатов с цифровыми подписями. Эти сертификаты предоставляются удостоверяющими центрами (англ. Certificate Authority).
- 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.

Структура сертификата X.509

Что такое OpenSSL

- OpenSSL — это криптографическая библиотека со свободным и открытым исходным кодом, которая предоставляет несколько инструментов командной строки для работы с цифровыми сертификатами. Некоторые из этих инструментов могут быть использованы в качестве центра сертификации.
- Центр сертификации (CA) является объектом, который подписывает цифровые сертификаты. Многие веб-сайты нуждаются в том, чтобы их клиенты знали, что соединение является безопасным, поэтому они платят международным доверенным центрам сертификации (например, VeriSign, DigiCert) за подпись сертификата для своего домена.
- В некоторых случаях имеет больше смысла выступать в качестве вашего собственного центра сертифкации, чем платить центрам сертификации, таким как DigiCert. Например, обеспечение безопасности вебсайта в интранете или для выдачи собственных сертификатов клиентам, чтобы позволить им проверять подлинность сервера (Apache, OpenVPN).

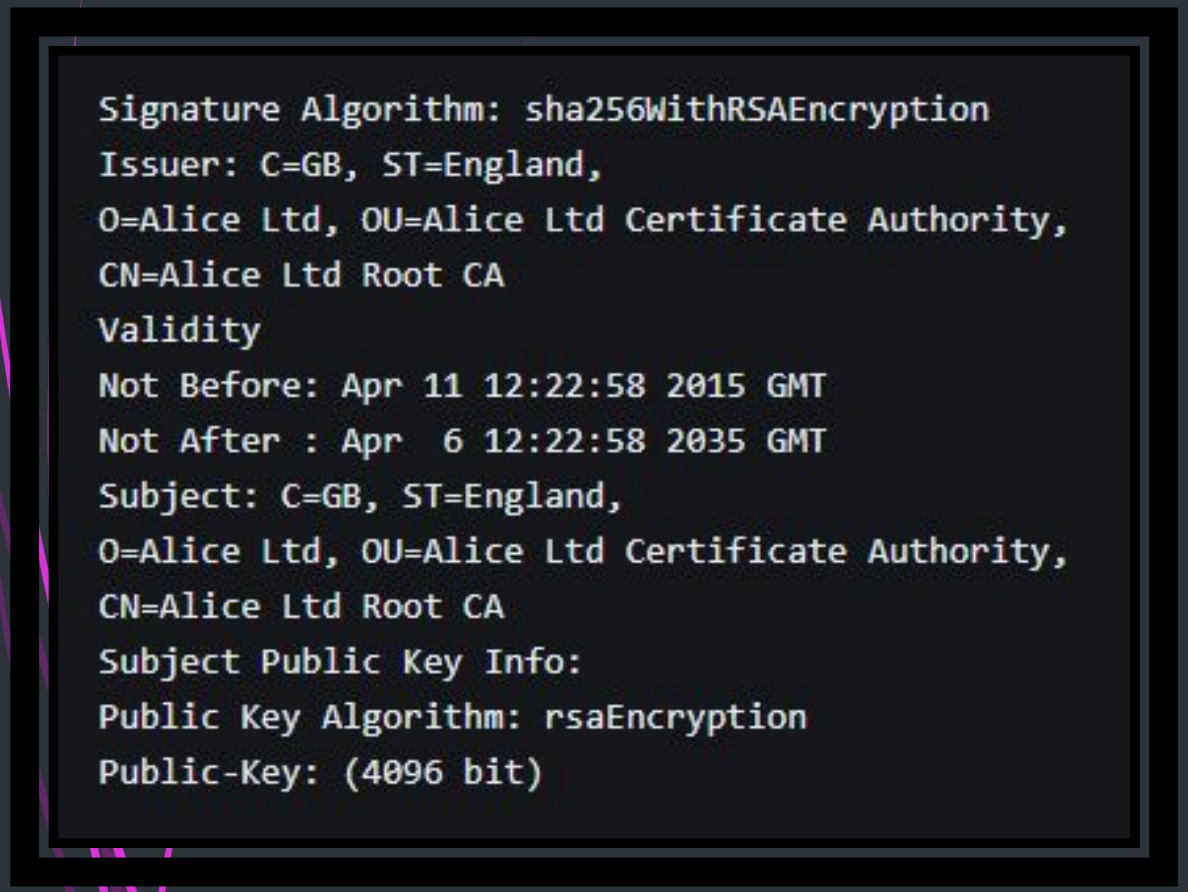
Цепочка доверия



- В компьютерной безопасности цифровые сертификаты проверяются с помощью цепочки доверия. Главным эмитентом в цепочке доверия является корневой центр сертификации (англ. root CA).
- Иерархия сертификатов — это структура, которая позволяет людям проверять валидность сертификата эмитента. Сертификаты подписываются закрытыми ключами тех сертификатов, которые находятся выше в иерархии сертификатов. Поэтому достоверность данного сертификата определяется достоверностью сертификата, которым он был подписан. Наивысший сертификат в цепочке называется корневым (Root certificate).

Создание сервера генерации сертификатов

- Выступать в качестве центра сертификации (CA) означает иметь дело с криптографическими парами приватных ключей и публичными сертификатами.
- Как правило, корневой центр сертификации не подписывает напрямую серверные или клиентские сертификаты. Корневой центр сертификации используется только для создания одного или нескольких промежуточных центров сертификации, которые являются доверенными корневому центру сертификации чтоб подписывать сертификаты от их имени
- Любой, владеющий корневым ключом, может выдавать доверенные сертификаты
- Для создания корневого сертификата используется корневой ключ.
- После того, как истечет корневой сертификат, все сертификаты, подписанные центром сертификации становятся недействительными.



Сертификат содержит в себе следующую информацию:

- Используемый алгоритм *Signature Algorithm*
- Даты периода действия сертификата *Validity*
- Длину публичного ключа *Public-Key*
- Эмитент сертификата *Issuer*, который является объектом, который подписал сертификат
- Предмет *Subject*, который относится к самому сертификату.
- Эмитент (*Issuer*) и предмет (*Subject*) идентичны для самоподписанных сертификатов. Все корневые сертификаты являются самоподписанными.

- Также в сертификате могут быть отображены расширения стандарта (v2 и v3)

```
X509v3 extensions:
X509v3 Subject Key Identifier:
38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31
X509v3 Authority Key Identifier:
keyid:38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31
X509v3 Basic Constraints: critical
CA:TRUE
X509v3 Key Usage: critical
Digital Signature, Certificate Sign, CRL Sign
```

Создание промежуточного центра сертификации

- Промежуточным центром сертификации является объект, который может подписывать сертификаты от имени корневого центра сертификации.
- Корневой центр сертификации подписывает промежуточный сертификат, образуя цепочку доверия.
- Цель использования промежуточного центра сертификации, прежде всего, для обеспечения безопасности.

Последовательность действий при создании промежуточного центра сертификации:

- Создается промежуточный ключ
- С использованием промежуточного ключа создается запрос на подписывание сертификата (CSR – certificate signing request)
- Через корневой центр сертификации, используя расширение стандарта X.509 для промежуточного CA, подписывается промежуточный запрос.

Промежуточный сертификат должен быть действителен на меньший период, чем корневой.

Цепочка сертификатов

- Когда приложение (например, веб-браузер) пытается проверить сертификат, подписанный промежуточным центром сертификации, оно также должно проверить промежуточный сертификат от корневого центра сертификации.
- Для выполнения проверки цепочки доверия создается цепочка сертификатов центра сертификации для предоставления приложениям.
- Чтобы создать цепочку сертификатов требуется объединить промежуточные и корневые сертификаты вместе.

Подписывание серверного и клиентского сертификатов

- Сертификаты подписываются через наш промежуточный центр сертификации.
- Эти подписанные сертификаты можно использовать в различных ситуациях, таких как защищать соединения к веб серверу или аутентифицировать клиентов, присоединяющихся к сервису.
- Стороннее лицо, однако, вместо этого может создать свой собственный частный ключ и запрос на подпись сертификата (CSR) без раскрытия вам своего частного ключа. Они дают вам собственный CSR, а вы отдаете им подписанный сертификат.

Создание серверного ключа

- Серверный и клиентский сертификат обычно истекают после одного года, поэтому мы можем безопасно использовать 2048 бит.
- Хотя 4096 бит немного больше безопаснее, чем 2048 бит, он замедляет TLS хэндшейки и значительно увеличивает нагрузку на процессор во время хэндшейков.
- Если вы создаете криптографическую пару для использования веб-сервером, вам потребуется вводить пароль каждый раз, когда вы перезапускаете веб-сервер. Вы можете указать опцию –aes256 для создания ключа без пароля.

```
# openssl genrsa -aes256 \
-out intermediate/private/www.example.com.key.pem 2048
```

Создание серверного сертификата

- Используя частный ключ, создается запрос на подписывание сертификата (CSR) промежуточному CA
- Сертификаты обычно даются сроком на один год, на центре сертификации обычно дают несколько дней для удобства.
- Используя файл цепочки сертификатов центра сертификации, который мы создали ранее, проводится проверка того, что новый сертификат имеет действительную цепочку доверия.

Установка сертификата

- Теперь можно установить новый сертификат на сервере, или распространить сертификат на клиентов. При установке на серверное приложение, понадобятся следующие файлы:
  - Файл цепочки сертификатов
  - Ключ сервера
  - Сертификат сервера
- Если вы подписывали CSR от стороннего лица, у вас нет доступа до их частного ключа, и вы должны им дать только файл цепочки и сертификат.

Списки отзывов сертификатов

- Списки отзывов сертификатов (CRL) предоставляют список сертификатов, которые были отозваны.
- Клиентское приложение, к примеру, веб-браузер, может использовать CRL для проверки подлинности сервера.
- Публикация CRL в публично доступном месте позволит третьим сторонам получать CRL из этого места, чтобы проверить, нет ли в нем сертификатов, которые могут быть отозваны.
- Некоторые разработчики приложений вместо устаревшего CRL используют Онлайн Протокол Статуса Сертификата (Online Certificate Status Protocol — OCSP).

Отзыв сертификата

- Обычно в центре сертификации, удостоверяющем клиентский сертификат, хранится файл ***index.txt***, содержащий в себе информацию обо всех сертификатах, подписанных этим центром, в виде:

```
V 160420124740Z 1001 unknown ... /CN=bob@example.com
```

- Однако, если сертификат клиента был отозван, соответствующая ему строка изменяется следующим образом:

```
R 160420124740Z 150411125310Z 1001 unknown ... /CN=bob@example.com
```

Использование CRL на сервере

- Обычно серверное приложение делает проверку для клиентских сертификатов. Это приложение должно иметь локальный доступ до CRL.
- В таком случае, сервер может проверять наличие записи о каждом клиентском сертификате в списке CRL и отказывать в доступе клиентам, чьи сертификаты были отозваны.

Использование CRL клиентами

- Для серверных сертификатов, обычно клиентское приложение (к примеру, веб-браузер) выполняет проверку. Это приложение должно иметь удаленный доступ к CRL.
- Если сертификат был подписан с правильным расширением, клиентское приложение может прочитать эту информацию и получить CRL из указанного места.
- Точки распространения CRL видны в спецификациях X509v3 сертификата.

```
# openssl x509 -in cute-kitten-pictures.example.com.cert.pem -noout -text
X509v3 CRL Distribution Points:
Full Name:
URI:http://example.com/intermediate.crl.pem
```

Online Certificate Status Protocol (OCSP)

- OCSP был создан в качестве альтернативы CRL.
- Как и CRL, OCSP позволяет запрашивающей стороне (к примеру, веб-браузеру) определять статус отзыва сертификата.
- Когда центр сертификации подписывает сертификат, он обычно включает адрес сервера OCSP в сертификат
- когда веб-браузеру предоставлен сертификат сервера, он посылает запрос на адрес сервера OCSP, указанном в сертификате. По этому адресу OCSP слушает запросы и отвечает статусом отзыва сертификата.
- Некоторыми веб-браузерами поддержка CRL считается устаревшей, или вообще убрана.

Список использованных источников

1. X.509 – URL: https://ru.wikipedia.org/wiki/X.509 (дата обращения 2022-04-19)
2. OpenSSL – URL: https://ru.wikipedia.org/wiki/OpenSSL#%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B8 (дата обращения 2022-04-20)
3. OpenSSL Certificate Authority. Центр сертификации OpenSSL. – URL: https://sgolubev.ru/openssl-ca/ (дата обращения 2022-04-20)
4. Основы HTTPS, TLS, SSL. Создание собственных X.509 сертификатов. Пример настройки TLSv1.2 в Spring Boothttps – URL: https://habr.com/ru/post/593507/ (дата обращения 2022-04-21)
5. OpenSSL: принципы работы, создание сертификатов, аудит – URL: https://hackware.ru/?p=12982 (дата обращения 2022-04-21)
6. Руководство. Создание тестовых сертификатов с помощью скриптов, предоставляемых корпорацией Майкрософт – URL: https://docs.microsoft.com/ru-ru/azure/iot-hub/tutorial-x509-scripts (дата обращения 2022-04-24)

Спасибо за внимание!