

Шпоры по сертификатам X.509

* [Информационная безопасность](https://habr.com/ru/hub/infosecurity/),
* [IT-инфраструктура](https://habr.com/ru/hub/it-infrastructure/),
* [IT-стандарты](https://habr.com/ru/hub/itstandarts/),
* [DevOps](https://habr.com/ru/hub/devops/)

*Чудище обло, озорно, огромно, стозевно и лаяй*.

Набор технологий, который мы по привычке именуем сертификатами SSL, представляет из себя здоровенный айсберг, на вершине которого зеленый замочек слева от доменного имени в адресной строке вашего браузера. Правильное название X.509 сертификат, который восходит к X.500 стандарту ITU-T DAP (Directory Access Protocol). DAP не взлетел, в IETF его посчитали неудобным для использования со всеми этими OSI нагромождениями и вместо него придумали LDAP, Lightweight DAP где первая буква обозначает «легковесный». Те, кому пришлось настраивать, или что хуже производить его отладку могут оценить иронию в полной мере. Никогда еще первая буква аббревиатуры так не лгала, не считая SNMP.



Кстати что общего между LDAP, SNMP и X.509 ну кроме того, что им еще не скоро предстоит собрать стадионы фанатов? Их объединяет **ASN.1** — мета-язык описания объектов древности. Если бы эти технологии создавали сейчас, в ход бы пошли *XML, DTD* или какой-нибудь другой ML. Но в то время стандарты создавались титанами, для которых даже SNMP был простым делом.

Словарный запас

Определение X.509 сертификатов есть в [архиве ITU-T](http://www.itu.int/rec/T-REC-X.509-200508-S)

Certificate ::= SEQUENCE {

tbsCertificate TBSCertificate,

signatureAlgorithm AlgorithmIdentifier,

signatureValue BIT STRING }

TBSCertificate ::= SEQUENCE {

version [0] EXPLICIT Version DEFAULT v1,

serialNumber CertificateSerialNumber,

signature AlgorithmIdentifier,

issuer Name,

validity Validity,

subject Name,

subjectPublicKeyInfo SubjectPublicKeyInfo,

issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,

*-- If present, version MUST be v2 or v3*

Для того, чтобы досконально понять обозначения и синтаксис, придется читать [спеки X.680 редакции 2008 г.](https://www.itu.int/rec/T-REC-X.680-201403-S!Cor2/en), где есть полное описание **ASN.1**. В понятиях **ASN.1** SEQUENCE обозначает примерно то же самое, что и struct в Си. Это может сбить с толку, ведь по семантике оно должно было соответствовать скорее массиву. И тем не менее.

Стандарт X.690 определяет следующие правила кодирования структур данных, созданных в соответствии с **ASN.1**: BER (Basic Encoding Rules), CER (Canonical Encoding Rules), DER (Distinguished Encoding Rules). Есть даже XER (XML Encoding Rules), [который](https://www.itu.int/en/ITU-T/asn1/Pages/xer.aspx) на практике мне никогда не встречался.

Да, но для чего нужны сертификаты X.509, которые доставляют столько головной боли? **Первая** и основная функция сертификатов X.509 — служить хранилищем *открытого или публичного ключа* PKI (public key infrastructure). К этой функции нареканий нет, а вот со второй не все так однозначно.

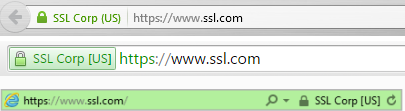
**Вторая** функция сертификатов X.509 заключается в том, чтобы предъявитель сего был принят человеком, либо программой в качестве истинного владельца некоего цифрового актива: доменного имени, веб сайта и пр. Это получается по-разному, далеко не все сертификаты имеют высокую ликвидность, если пользоваться финансовой терминологией. Полгода назад Гугл [пригрозил](https://goo.gl/i4ovo5) компании Симантек, что перестанет доверять их сертификатам из-за того, что те выпустили аж 30,000 неисправных сертификатов.

Номенклатура сертификатов

Давайте рассмотрим, какие сертификаты X.509 встречаются в природе, если рассматривать их по расположению в  цепочке доверия.

* **Корневые сертификаты** — изготовлены в корневом УЦ (удостоверяющий центр) и имеют следующие признаки: атрибуты issue и subject идентичны, а в расширении basicConstraints атрибут cA принимает значение TRUE.
* **Промежуточные сертификаты** — расплывчатый термин, обозначающий сертификаты не подписанные корневым УЦ, которые могут формировать цепочку произвольной длины, начиная от *корневого сертификата* и заканчивая *сертификатом конечного субъекта*.
* **Сертификаты конечного субъекта** — конечные сертификаты в цепочке, которые не могут подписывать другие *промежуточные сертификаты* своим закрытым ключом.

По степени  дороговизны и надежности сертификаты делятся на 3 вида: **DV**, **OV** и **EV**.

* **DV** — сертификаты удостоверения доменного имени получить проще простого. Они выдаются автоматически и моментально после того, как центр сертификации проверит, что заявитель имеет право на доменное имя. Чаще всего для этого достаточно открыть сообщение и перейти по указанной ссылке. Естественно, что сообщение будет отправлено на почтовый ящик с доменным именем, которое следует удостоверить.
* **OV** — в сертификате будет уже указано не доменное имя, а название самой организации заявителя. Тут уже ни а какой автоматической выдачи речи быть не может, это займет несколько рабочих дней. Проверке подлежит наличие в базе whois домена название организации заявителя. Могут проверить государственную регистрацию и валидность телефонного номера.
* **EV** — эти сертификаты и получить сложно и стоят они недешево. Их можно опознать по названию организации на зеленом замочке на панели адресной строки.  
    
  

Редко, кто на это готов раскошелиться. Навскидку Яндекс, StackOverflow.com и Хабр могут жить и без него. Но те, кто готов пойти ради этого на жертвы должны выполнить следующие требования:

1. Аудит правовой, физической и операционной деятельности организации.
2. Следует убедиться в том, что организация имеет эксклюзивное право на использование доменного имени.
3. Следует убедиться в том, что организация авторизована для выпуска сертификата данного типа.

Более подробно можно прочитать в [Хабрапоспе компании TutHost](https://habrahabr.ru/company/tuthost/blog/150433/). Также атрибут subject X.509 EV сертификата содержит значения jurisdictionOfIncorporationCountryName, businessCategory, и serialNumber.

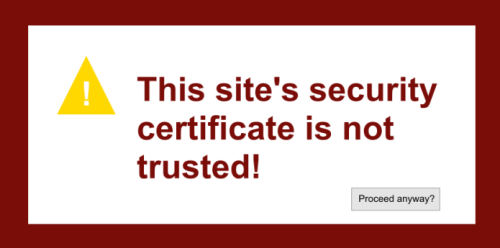
По свои свойствам сертификаты бывают следующих типов.

* **Мульти-доменные сертификаты** — сертификат может охватывать несколько доменных имен с помощью SAN — атрибута subjectAltName.
* **Мульти-хостовые сертификаты** — в тех случаях, когда атрибут subject содержит запись CN=example.net, в время как DNS сервер может иметь несколько записей A / AAAA типа, где одно имя узла может соответствовать нескольким IP адресам. В этом случае сертификат X.509 с одним и тем же hostname может быть успешно восстановлен на всех подобных узлах.
* **Сертификаты с возможностью подстановки, wildcard сертификаты** — это когда атрибут subject содержит запись CN=\*.example.net. Действует так же, как и в привычных регулярных выражениях, то есть может быть использован на всех под-доменах \*.example.net.
* **Квалифицированные сертификаты** — [RFC 3739](https://tools.ietf.org/html/rfc3739) определяет этот термин, как относящийся *персональным* сертификатам, ссылаясь на *Директиву Европейского Союза об электронной подписи*. В частности RFC позволяет в атрибуте subject содержать значения:
  + commonName (CN=),
  + givenName (GN=),
  + pseudonym=.  
      
    Также subjectDirectoryAttributes включает в себя значения:
  + dateOfBirth=,
  + placeOfBirth=,
  + gender=,
  + countryOfCitizenship=,
  + countryOfResidence=.

В России понятие КС *квалифицированного сертификата* [определено законодательно](http://r77.center-inform.ru/support/qualified-certificate/) в связи с доступом к ГосУслугам. По [ссыске](https://habrahabr.ru/post/325998/) Хабрапост с былиной об извлечении персональных данных из КС.

Откуда берутся сертификаты?

Еще совсем недавно было всего 2 способа заполучить X.509 сертификат, но времена меняются и с недавнего времени есть и третий путь.

1. Создать свой собственный сертификат и самому же его подписать. Плюсы — это бесплатно, минусы — сертификат будет принят лишь вами и, в лучшем случае, вашей организацией.  
     
   
2. Приобрести сертификат в УЦ. Это будет стоить денег в зависимости от различных его характеристик и возможностей, указанных выше.
3. [Получить](https://habrahabr.ru/post/270273/) бесплатный сертификат [LetsEncrypt](https://letsencrypt.org/), доступны только самые простые **DV** сертификаты.

Для первого сценария достаточно пары команд и чтобы 2 раза не вставать создадим сертификат с алгоритмом **эллиптических кривых**. Первым шагом нужно создать закрытый ключ. Считается, что шифрование с алгоритмом эллиптических кривых дает [больший выхлоп](https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/), если измерять в тактах CPU, либо байтах длины ключа. Поддержка ECC *не* определена однозначно в TLS < 1.2.

openssl ecparam -name secp521r1 -genkey -param\_enc explicit -out private-key.pem

Далее, создает CSR — запрос на подписание сертификата.

openssl req -new -sha256 -key private.key -out server.csr -days 730

И подписываем.

openssl x509 -req -sha256 -days 365 -in server.csr -signkey private.key -out public.crt

Результат можно посмотреть командой:

openssl x509 -text -noout -in public.crt

Openssl имеет огромное количество опций и команд. Man страница не очень полезна, справочник удобнее использовать так:

openssl -help

openssl x509 -help

openssl s\_client -help

Ровно то же самое можно сделать с помощью java утилиты keytool.

keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048

Следует серия вопросов, чтобы было чем запомнить поля owner и issuer

What is your first and last name?

What is the name of your organizational unit?

What is the name of your organization?

What is the name of your City or Locality?

What is the name of your State or Province?

What is the two-letter country code for this unit?

Is CN=Johnnie Walker, OU=Unknown, O=Unknown, L=Moscow, ST=Moscow, C=RU correct?

Конвертируем связку ключей из проприетарного формата в PKCS12.

keytool -importkeystore -srckeystore keystore.jks -destkeystore keystore.jks -deststoretype pkcs12

Смотрим на результат:

keytool -list -v -alias selfsigned -storepass password -keystore keystore.jks

Значению ObjectId: 2.5.29.14 соответствует определение ASN.1, согласно [RFC 3280](https://tools.ietf.org/html/rfc3280.html) оно всегда non-critical. Точно так же можно узнать смысл и возможные значения других ObjectId, которые присутствуют в сертификате X.509.

subjectKeyIdentifier EXTENSION ::= {

SYNTAX SubjectKeyIdentifier

IDENTIFIED BY id-ce-subjectKeyIdentifier

}

SubjectKeyIdentifier ::= KeyIdentifier

LetsEncrypt

Можно бесплатно получить X.509 сертификат *LetsEncrypt* и для этого не нужно даже заходить на вебсайт, достаточно установить certbot.

sudo emerge -av certbot *#для Gentoo*

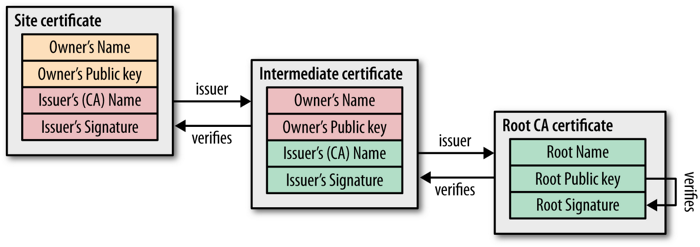
sudo apt-get install certbot -t stretch-backports *#Debian*

sudo dnf install certbot *#Fedora*

sudo certbot certonly --standalone -d example.com -d www.example.com

Сценарий №1 — найти следующего в связке

**Связка сертификатов** — Объединение нескольких X.509 сертификатов в один файл, чаще всего в формате PEM. Связка передается по сети в момент протокола рукопожатия SSL/TLS.



Самый сок начинается, когда имеете дело со связкой сертификатов, a. k. a certificate chain. Часто просматривая лапшу в связке ключей jks непросто понять как найти родительский сертификат, когда там россыпь новых и старых сертификатов на несколько доменных имен.

Рассмотрим связку сертификатов \*.novell.com. Расширение Authority Key Identifier (AKI) должно совпадать с Subject Key Identifier (SKI) старшего в связке.

Certificate Authority Key Identifier

Size: 20 Bytes / 160 Bits

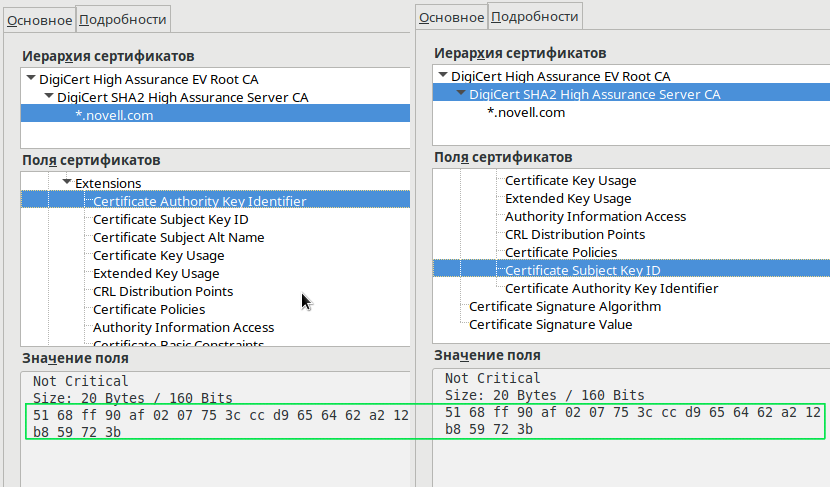
51 68 ff 90 af 02 07 75 3c cc d9 65 64 62 a2 12 b8 59 72 3b

Так и есть, SKI сертификат DigiCert имеет то же значение.

Certificate Subject Key ID

Size: 20 Bytes / 160 Bits

51 68 ff 90 af 02 07 75 3c cc d9 65 64 62 a2 12 b8 59 72 3b



Для корневого сертификата AKI = SKI, а также isCa=true

Certificate Basic Constraints

Critical

Is a Certificate Authority

Сценарий №2 — используй subjectAltnName, Люк

Вот представьте у вас приложение, использующее веб сервер: вики, WordPress или Cacti. Вы настроили доступ по https, приобрели или сами сгенерировали и подписали сертификат. Все должно быть в порядке, но зеленого замочка все равно нет. Браузер подозревает, что сертификат готовили неправильные пчелы, из-за того, что FQDN сервера и hostname, который указан в адресной строке не совпадают. Так иногда бывает, что DNS сервера указывает на mars.domain.com, а веб-сервер настроен на venus.domain.com.

Если администратору в силу перфекционизма нужны помимо езды нужны еще и шашечки — вожделенный зеленый замочек, то нужно переделать сертификат X.509, определив в нем subjectAltName.

Откройте файл openssl.cnf и в секции req добавьте следующие линии.

[ alternate\_names ]

DNS.1 = example.com

DNS.2 = www.example.com

DNS.3 = mail.example.com

DNS.4 = ftp.example.com

Далее, в секции [ v3\_ca ] укажите.

subjectAltName = @alternate\_names

А дальше все как обычно, создаем закрытый ключ и подписываем сертификат.

openssl genrsa -out private.key 3072

openssl req -new -x509 -key private.key -sha256 -out certificate.pem -days 730

Использованные материалы

* [Survival guides — TLS/SSL and SSL (X.509) Certificates](http://zytrax.com/tech/survival/ssl.html#x509-usage)
* [Разбираем x.509 сертификат](https://habrahabr.ru/post/194664/)
* [Creating ECDSA SSL Certificates in 3 Easy Steps](https://zonena.me/2016/02/creating-ssl-certificates-in-3-easy-steps/)
* [Implementing SSL/TLS Using Cryptography and PKI](https://www.safaribooksonline.com/library/view/implementing-ssltls-using/)
* [Certificate Chaining Engine — how it works](https://www.sysadmins.lv/blog-en/certificate-chaining-engine-how-this-works.aspx)
* [How can I generate a self-signed certificate with SubjectAltName using OpenSSL? [closed]](https://stackoverflow.com/questions/21488845/how-can-i-generate-a-self-signed-certificate-with-subjectaltname-using-openssl)

**Теги:**

* [x.509](https://habr.com/ru/search/?q=%5Bx.509%5D&target_type=posts)
* [openssl](https://habr.com/ru/search/?q=%5Bopenssl%5D&target_type=posts)
* [keytool](https://habr.com/ru/search/?q=%5Bkeytool%5D&target_type=posts)