SSL Howto

Введение

Большинство сетевых протоколов, например, IMAP, POP, SMTP, HTTP, FTP и LDAP, обеспечивают поддержку шифрования информации по протоколу SSL. Обычно шифрование используется для скрытия передаваемых по сетям общего пользования логинов и паролей доступа к различным ресурсам, например, средствам администрирования через Web-интерфейс, акаунтов для предоставления различных услуг (хостинг, электронная почта, управление мобильным телефоном и т. п.), почтовым ящикам, закрытым каталогам на FTP и Web-серверах для передачи различных конфиденциальных сообщений. При передаче аутентификационной информации в виде обычного текста она может быть перехвачена третьими лицами с использованием программ-сниферов и использована для получения несанкционированного доступа к вашим ресурсам.

В некоторых странах ввоз, распространение и использование программного обеспечения для криптографии запрещено. Использование алгоритмов RC5 и IDEA, реализованных в OpenSSL, требует получения соответствующих лицензий.

Немного теории

В настоящее время протокол SSL практически незаметно для пользователя взаимодействует с остальными протоколами Интернет и обеспечивает передачу конфиденциальной информации по сетям общего пользования в зашифрованном виде. Программное обеспечение OpenSSL поддерживает протоколы SSL (Secure Sockets Layer) v2/v3 и TLS v1 (Transport Layer Security). Утилита командной строки openssl может использоваться для выполнения следующих задач:

- Создание и управление ключами RSA и DSA
- Создание сертификатов формата x509, запросов на подпись сертификатов, восстановление сертификатов
- Шифровать данные с помощью симметрического или асимметрического алгоритма шифрования
- Работать с S/MIME
- Производить ssl/tls тестирование серверов и клиентов

SSL использует асимметричные алгоритмы шифрования, т.н. шифрование с открытым ключом. В таком алгоритме один ключ используется для расшифровки данных, зашифрованных другим ключом. Эти два ключа (private/public) - пара ключей - создаются одновременно, с помощью математической формулы. Это единственный простой способ найти два ключа, обладающих специальной ассиметрией (при которой один из ключей может расшифровать зашифрованное вторым): вычисление одного из ключей по второму - задача невероятно сложная. При применении шифрования с открытым ключом один ключ из пары делается свободно доступным, т.н. public key, а второй сохраняется в секрете, т.н. private key. Тот, кто хочет установить с создателем ключей безопасный контакт, может перед отправкой зашифровать свое сообщение, пользуясь открытым ключом (public key). Если адресат хранит закрытый ключ (private key) в секрете, то только он сможет расшифровать сообщение.

И наоборот, автор пары ключей может зашифровать свое сообщение закрытым ключом. Получатель может удостовериться, что сообщение действительно принадлежит автору ключей, попробовав расшифровать его с помощью открытого ключа. Если сообщение после расшифровки имеет какой то смысл, а отправитель действительно сохранил зарытый ключ в секрете, то сообщение действительно написано им. Успешная расшифровка также подтверждает, что сообщение не было изменено в процессе доставки (например, при прохождении через почтовый сервер), поскольку в противном случае не была бы получена правильная расшифровка. Так сообщение проверяется получателем.

К сожалению, шифрование больших объемов данных с помощью ассиметричных алгоритмов происходит медленно - гораздо медленнее, чем при использовании симметричных алгоритмов. Но при использовании шифрования с открытым ключом, в целях проверки подлинности (а не для сохранения конфиденциальности), совершенно не обязательно шифровать сообщение целиком. Вместо этого сообщение сначала обрабатывается вычислительно необратимой хеш-функцией, например с использованием алгоритма md5. Затем достаточно зашифровать только хеш-значение, которое является представлением данных. Зашифрованное хеш-значение, называемое теперь цифровой подписью, добавляется к сообщению, для которого будет производиться проверка подлинности. Получатель может проверить подлинность сообщения, расшифровав подпись и обработав сообщение копией вычислительно необратимой хеш-функции. Если полученные хеш-значения совпадают - сообщение подлинное.

Вычислительно необратимая хеш-функция представляет собой специальный вид математических формул. Эта хеш-функция, известная также под именем криптографической контрольной суммы или дайджеста сообщения, вычисляет хеш-значение фиксированной длины на основе исходных данных произвольного объема.

Суть вычислительно необратимой хеш-функции заключается в том, что каждый бит хеш-значения зависит от каждого из битов исходных данных. Если изменить единственный бит исходных данных, хеш-значение тоже изменится - очень сильно и непредсказуемо - настолько непредсказуемо, что задача обращения функции и нахождения ввода, приведшего к получению конкретного хеш-значения, является «вычислительно неосуществимой».

Сертификат содержит публичный ключ, подписанный одним из корневых доверенных центров сертификации, данные об организации, выдавшей сертификат и в некоторых случаях зашифрованный закрытый ключ, а также отпечаток (хеш) публичного ключа.

Сертификаты имеют время действия, по окончанию которого они автоматически считаются недействительными, иерархия сертификатов обычно строится на основании сети доверия (бывают довольно длинные цепочки сертификатов, ведущие к доверенному ключу из root CA).

Таким образом, сертификат - это полный комплекс системы асимметрического шифрования, предоставляющий гораздо больше возможностей, чем сами по себе ключи (а также являющийся более защищенной системой). Одним из основных достоинств сертификата является возможность записи в него информации об организации, этот ключ выдавшей.

Сертификаты играют важную роль в коммуникационном процессе. Сертификат, подписанный доверенным CA, дает гарантию, что владелец сертификата, тот за кого он себя выдает. Без подписанного сертификата ваши данные будут зашифрованы, однако, сервер, с которым вы контактируете, может быть не тем, за кого себя выдает.

Подготовительная часть

Итак, для начала создаем директорию, где мы будем работать. Например, можно создать ее в своей домашней папке.

```
# cd ~
# mkdir CA
# cd CA
# mkdir newcerts private
```

Папка CA будет содержать сертификат нашего CA, базу данных сертификатов, которые мы подписали, а также ключи, запросы и сертификаты, которые мы сгенерируем. Она также будет нашей рабочей директорией, когда мы будем создавать или подписывать сертификаты.

- CA/newcerts - будет содержать копию каждого сертификата, который мы подпишем.
- CA/private - будет содержать наш CA private key.

Этот ключ (CA private key) очень важен. Не теряйте этот ключ, без него, вы не сможете подписать или обновить сертификаты. Не показывайте этот ключ никому, если его узнают, то злоумышленник сможет выдавать себя за вас.

Наш следующий шаг заключается в создании БД для сертификатов, которые мы будем подписывать.

```
# echo '01' > serial
# touch index.txt
```

Вместо того, чтобы использовать конфигурационный файл, поставляемый вместе с openssl (**/etc/ssl/openssl.cnf**), мы создадим свой, с минимально необходимой конфигурацией. Этот файл очень удобен, т.к. при создании или подписи сертификатов или запросов нам придется каждый раз вводить много одной и той же информации. Вместо этого мы можем указать openssl брать информацию из конфигурационного файла.

```
#
# /root/CA/openssl.cnf
#

# Задаем рабочую директорию
dir      = .
```

При использовании OpenSSL большая часть того, что будет включаться в сертификат, зависит от настроек конфигурационного файла (openssl.cnf), а не от параметров командной строки.

Конфигурационный файл разделен на секции, которые выборочно читаются и обрабатываются согласно аргументам командной строки openssl. Каждая секция может включать одну или несколько других секций, ссылаясь на них, что позволяет сделать конфигурационный файл модульным. Имя в квадратных скобках (**[название]**) означает начало каждой секции.

Создание корневого (самоподписанного) сертификата

Создаем корневой сертификат, с помощью которого будем подписывать все наши сертификаты. Сейчас нам необходимо добавить секцию, которая управляет процессом создания сертификатов, а также секцию, которая определяет тип создаваемых сертификатов. Первое, что нам нужно это указать Distinguished Name. Это текст, определяющий владельца сертификата, при его просмотре.

Добавьте следующие строки в ваш openssl.cnf

```
# В данной секции описываются основные опции
[ req ]

# Длина ключа в битах
default_bits = 4096

# Алгоритм шифрования
default_md = md5

# Разрешенные символы
string_mask = nombstr

# Указываем, что DN (Distinguished Name) будет описана в секции
req_distinguished_name
distinguished_name = req_distinguished_name

# В данной секции указываются данные, которые будут использоваться
# по умолчанию при генерации запроса на подписание сертификата
[ req_distinguished_name ]
0.organizationName       = Organization Name (company)
organizationalUnitName   = Organizational Unit Name (department, division)
emailAddress             = Email Address
emailAddress_max         = 40
localityName             = Locality Name (city, district)
stateOrProvinceName      = State or Province Name (full name)
countryName              = Country Name (2 letter code)
countryName_min          = 2
countryName_max          = 2
commonName               = Common Name (hostname, IP, or your name)
commonName_max           = 64

# Значения по умолчанию
0.organizationName_default    = Turbogaz
localityName_default          = Kharkov
stateOrProvinceName_default   = Ukraine
countryName_default           = UA

[ v3_ca ]
basicConstraints         = CA:TRUE
subjectKeyIdentifier     = hash
```

```
authorityKeyIdentifier   = keyid:always,issuer:always
```

Чтобы защитить себя, от попытки несанкционированного использования нашего CA сертификата мы защитим его паролем. Каждый раз, когда вы будете использовать CA сертификат для подписи запроса, у вас будут спрашивать этот пароль.

В целях безопасности пароль лучше сгенерировать с помощью специальной программы, чтобы он отвечал требованиям безопасности (длина не менее 12 символов, использование символов верхнего и нижнего регистра, использование цифр и спецсимволов и т.д.). Только потом не забудьте этот пароль ;).

Итак, у нас все готово для создания нашего самоподписанного корневого сертификата.

```
# openssl req -new -x509 -extensions v3_ca -keyout private/rootCA.key -out
rootCA.crt -days 1825 -config ./openssl.cnf
Generating a 4096 bit RSA private key
...
...
...
writing new private key to 'private/rootCA.key'
Enter PEM pass phrase:*******
Verifying - Enter PEM pass phrase:*******
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Organization Name (company) [Turbogaz]:
Organizational Unit Name (department, division) []:Turbogaz Security
Division
Email Address []:security@turbogaz.net
Locality Name (city, district) [Kharkov]:
State or Province Name (full name) [Ukraine]:
Country Name (2 letter code) [UA]:
Common Name (hostname, IP, or your name) []:Turbogaz Root CA
```

Создать новый [-new], самоподписанный сертификат [-x509] для использования в качестве сертификата CA [-extensions v3_ca], а именно basicConstraints = CA:TRUE. Сертификат создается с использованием секретного ключа, который будет сохранен в rootCA.key [-keyout private/rootCA.key] и конфигурационного файла openssl.cnf [-config ./openssl.cnf]. Созданный сертификат будет называться rootCA.crt [-out rootCA.crt] и будет действителен в течение 1825 дней [-days 1825].
Когда время действия корневого сертификата истекает, все подписанные им сертификаты становятся недействительными. Для исправления этой ситуации, необходимо создать и распространить новый корневой сертификат.

Также необходимо отозвать и заново подписать все сертификаты, подписанные просроченным корневым сертификатом. Это может занять очень много времени, поэтому вы, возможно,

захотите указать большой период действия корневого сертификата (в нашем случае 1825 дней ~ 5 лет)

Поэтому у доверенных корневых центров сертификации период действия корневого сертификата, как правило, не менее 10 лет.

Просмотреть информацию о назначении сертификата можно с помощью следующей команды

```
# openssl x509 -in rootCA.crt -noout -purpose
Certificate purposes:
SSL client : Yes
SSL client CA : Yes
SSL server : Yes
SSL server CA : Yes
Netscape SSL server : Yes
Netscape SSL server CA : Yes
S/MIME signing : Yes
S/MIME signing CA : Yes
S/MIME encryption : Yes
S/MIME encryption CA : Yes
CRL signing : Yes
CRL signing CA : Yes
Any Purpose : Yes
Any Purpose CA : Yes
OCSP helper : Yes
OCSP helper CA : Yes
```

Просмотреть информацию о периоде действия сертификата можно с помощью следующей команды

```
# openssl x509 -in rootCA.crt -noout -dates
notBefore=May  8 15:44:20 2006 GMT
notAfter=May  7 15:44:20 2011 GMT
```

Просмотреть информацию о сертификате можно с помощью следующей команды

```
# openssl x509 -in rootCA.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            f9:3c:b0:9d:41:6a:37:61
        Signature Algorithm: md5WithRSAEncryption
        Issuer: O=Turbogaz, OU=Turbogaz Security Division/
        emailAddress=security@turbogaz.net, L=Kharkov, ST=Ukraine, C=UA,
        CN=Turbogaz Root CA
        Validity
            Not Before: May  8 15:44:20 2006 GMT
            Not After : May  7 15:44:20 2011 GMT
        Subject: O=Turbogaz, OU=Turbogaz Security Division/
```

```
        emailAddress=security@turbogaz.net, L=Kharkov, ST=Ukraine, C=UA,
        CN=Turbogaz Root CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (4096 bit)
                Modulus (4096 bit):
                    00:ed:34:f1:54:ea:f4:0f:44:c8:58:22:78:6e:10:
                    70:d5:c0:bd:2f:ce:bb:ab:ac:85:21:a3:3f:e3:c4:
                    9c:ae:c6:09:82:92:82:9e:6e:50:08:d0:56:aa:ba:
                    79:5a:76:eb:db:f1:e8:d1:3c:0b:c9:ab:4d:c0:cc:
                    d0:c1:ab:73:d5:25:86:98:99:03:e5:9b:0d:26:7b:
                    b2:91:95:f3:da:84:1a:f9:d9:65:a9:f3:2f:0e:76:
                    dd:af:f7:cc:c9:3c:f3:3a:26:5e:64:66:97:75:be:
                    9d:8b:a8:6a:9c:ea:53:43:c8:97:52:d8:82:73:0c:
                    c2:98:9d:1a:6b:c5:6b:4e:73:07:29:0e:b4:cf:ef:
                    af:38:de:f8:31:ac:2f:d1:a4:aa:36:ae:13:8b:5c:
                    1f:e5:0f:83:c8:46:8c:d8:f9:be:3f:88:0a:5d:72:
                    20:0a:33:da:78:f8:a6:f4:40:30:7c:c1:8b:c6:00:
                    25:b8:ba:7c:fe:16:f0:36:c9:f2:e3:cb:19:26:b7:
                    db:cf:6e:dd:f9:2b:f5:11:82:e4:fc:6b:9d:2f:57:
                    e6:ce:42:d9:bb:4d:bb:1b:38:97:4b:67:c7:a9:12:
                    e0:6d:e0:5d:e7:36:62:fc:9a:b5:99:75:7e:d9:68:
                    41:23:44:f9:46:cf:c5:b5:3d:3b:5f:d7:cc:85:6a:
                    39:68:26:c4:95:e9:31:68:fd:5d:f9:c6:b3:a7:bd:
                    13:a8:17:db:f8:e5:42:99:52:57:dc:de:1d:9a:88:
                    7f:af:99:99:bb:e6:9b:5f:fc:e0:5c:4e:71:9b:2f:
                    50:7e:d9:80:9e:5d:d3:a2:85:4f:2f:b0:5f:a1:ae:
                    60:2b:4b:42:ae:cb:af:ef:6e:2e:2c:b5:70:69:a2:
                    88:d5:a1:3e:ff:de:3a:8a:38:16:6c:57:6a:a5:b7:
                    de:1b:f3:8a:7f:45:2a:8d:9e:c8:c3:0c:b2:8c:bd:
                    76:01:ba:82:99:52:ad:82:b5:34:25:52:07:74:5c:
                    69:a5:2a:ff:ed:b8:28:96:4e:4e:58:24:26:0f:15:
                    6f:d3:97:de:87:84:b8:12:84:35:eb:ee:94:80:1d:
                    00:6a:3e:6b:8b:9e:66:e6:d5:ee:3d:98:69:40:6c:
                    f4:97:6b:46:ae:3b:b3:36:8c:e3:45:b5:8f:9f:58:
                    93:ff:48:eb:27:5a:63:f4:9a:f3:72:f9:c3:26:92:
                    01:b2:bb:5d:ea:a9:d0:31:31:7b:6a:d4:2e:d5:e9:
                    7f:02:4c:03:f6:6c:8c:0b:57:90:b6:cc:f0:e4:ca:
                    29:85:8a:99:33:fd:a4:90:4a:58:2d:91:49:d1:26:
                    8c:ea:e6:d0:bc:5c:a6:03:5d:84:bb:37:56:f3:6b:
                    9f:df:57
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                5F:70:27:E5:B0:80:59:C9:1E:CC:94:5F:62:BD:54:03:29:B9:08:BF
            X509v3 Authority Key Identifier:
keyid:5F:70:27:E5:B0:80:59:C9:1E:CC:94:5F:62:BD:54:03:29:B9:08:BF
                DirName:/O=Turbogaz/OU=Turbogaz Security Division/
emailAddress=security@turbogaz.net/L=Kharkov/ST=Ukraine/C=UA/
```

```
                CN=Turbogaz Root CA
                serial:F9:3C:B0:9D:41:6A:37:61

    Signature Algorithm: md5WithRSAEncryption
        5c:ea:dd:f4:1b:fc:26:45:97:e9:5d:fc:5e:4d:8c:45:aa:6d:
        4d:6f:43:60:af:e3:db:e7:eb:57:68:85:b9:0d:b4:8d:46:32:
        63:0f:bd:88:11:44:d2:9b:36:c4:13:cc:53:6d:1c:cb:b9:72:
        fc:da:8a:f2:b0:92:d5:61:3f:1d:8f:59:14:aa:3b:8a:e2:b3:
        84:24:42:ef:c1:fb:70:b3:77:88:6d:c3:b8:23:76:dd:98:0a:
        98:97:33:44:51:7a:b6:66:05:ad:c9:64:8a:c2:43:3c:21:15:
        43:69:21:02:2d:0a:d5:86:d9:bf:29:43:ea:ae:d7:bd:5c:d1:
        36:e4:7e:22:ba:a6:f6:d0:5e:f8:73:a4:45:96:83:5f:a5:e3:
        91:76:e7:33:a6:c1:34:c5:67:e6:95:81:45:ac:85:dd:82:9d:
        a2:a8:62:ef:c0:c4:80:df:f7:49:5a:6b:68:dd:f8:00:47:31:
        d8:67:60:2a:65:41:b2:e3:3b:53:07:27:88:88:ed:89:20:5e:
        eb:f3:f5:0a:d2:02:cc:12:9c:5d:c6:37:37:4f:f8:7a:2c:52:
        3c:c1:68:cb:54:9b:de:51:27:08:95:67:c7:b0:6a:c8:a7:0a:
        3c:d4:3a:07:4a:8b:6a:a9:d1:a0:76:8f:63:09:04:91:dd:bb:
        7b:22:42:de:c8:07:ef:4a:05:8f:44:3d:4c:24:b4:e2:86:59:
        b1:f6:14:6a:7f:b2:e0:02:34:ce:d0:2b:cf:ee:4a:3f:23:4a:
        ae:2c:3d:7c:72:6f:c1:d0:a8:56:3a:83:f4:1a:08:c4:5a:21:
        58:2e:23:f0:ff:bb:61:15:23:c8:71:40:33:25:5f:66:80:ba:
        a4:64:b9:aa:4a:ae:ce:e6:7f:21:64:38:16:81:27:f1:0c:95:
        26:d7:8a:d1:9c:e0:da:70:6b:bb:9f:bc:0a:7e:2f:eb:23:b5:
        95:ec:04:0a:47:ae:fc:04:7d:2f:12:c6:a6:94:24:88:cc:09:
        bd:c4:10:8d:48:16:ec:9e:57:07:12:97:77:b8:65:d2:e9:95:
        8b:aa:d8:94:45:24:f5:4f:ec:a8:4a:03:7d:82:35:2b:36:80:
        90:2a:26:31:35:40:b6:ea:8e:72:e6:7c:06:91:c2:4a:77:a3:
        fc:91:bc:ad:9d:d3:ff:59:8f:35:65:e3:d4:20:f1:c4:b4:db:
        a4:3d:f2:1e:ee:d2:80:92:e8:f4:f3:35:10:0e:dd:39:51:cf:
        04:be:ba:14:6f:c3:f4:bd:e6:79:51:d5:d2:b6:c1:74:bb:7e:
        1e:27:8c:ee:8a:51:a3:a8:5e:8b:fc:c3:e7:78:65:b3:83:31:
        fd:85:42:3a:df:b0:5d:73
```

В результате выполнения этой команды мы получили два файла:

- private/rootCA.key - закрытый ключ
- rootCA.crt - корневой CA сертификат

rootCA.crt - это файл, который вы должны разослать вашим клиентам. Чтобы они добавили его в список доверенных корневых сертификатов. rootCA.key - этот файл вы не должны никому показывать.

Создание запроса на подпись сертификата

Теперь, когда мы создали корневой сертификат, мы можем создать любое количество сертификатов для установки в ssl приложения, такие как http, ftp, pop, imap, stunnel и т.д.

Эта процедура состоит из создания закрытого ключа и запроса на подпись сертификата, а

далее в подписи его нашим корневым сертификатом.

Для создания не корневых сертификатов нам необходимо внести изменения в наш конфигурационный файл. Добавьте следующие строчки в конец файла.

```
[ v3_req ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
```

Чтобы часто не вводить данные в командную строку, добавьте следующую строку в секцию `[ req ]` после параметра `distinguished_name`.

```
req_extensions    = v3_req
```

Теперь мы готовы создать наш первый запрос на сертификат. В этом примере мы создадим сертификат для почтового сервера. Все шаги практически такие же, как и при создании CA сертификата. Но три поля имеют другое назначение, а именно:

- Organizational Unit - необходимо указать, для чего предназначен сертификат
- Email Address - postmaster@turbogaz.net
- Common Name - fqdn или ip адрес почтового сервера

```
# openssl req -new -nodes -out smtp.turbogaz.net.csr -keyout
smtp.turbogaz.net.key -config ./openssl.cnf
Generating a 4096 bit RSA private key
...
...
...
writing new private key to 'smtp.turbogaz.net.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Organization Name (company) [Turbogaz]:
Organizational Unit Name (department, division) []:Turbogaz SMTP Server
Email Address []:postmaster@turbogaz.net
Locality Name (city, district) [Kharkov]:
State or Province Name (full name) [Ukraine]:
Country Name (2 letter code) [UA]:
Common Name (hostname, IP, or your name) []:smtp.turbogaz.net
```

В результате выполнения этой команды мы получили два файла:

- smtp.turbogaz.net.key - секретный закрытый ключ
- smtp.turbogaz.net.csr - запрос на подпись сертификата

Эти файлы необходимо сохранить. Когда срок действия сертификата истечет, то запрос может быть использован снова для создания нового сертификата с новым сроком действия. Закрытый ключ конечно необходим для SSL кодирования.

Подпись сертификата

Теперь нам необходимо добавить в наш конфигурационный файл секцию, которая отвечает за CA. Эта секция определяет пути к различным частям, таким как база данных, CA сертификат и закрытый ключ. Она также содержит некоторые базовые настройки. Добавьте следующие строчки в файл openssl.cnf перед секцией **[req]**.

```
[ ca ]
default_ca    = CA_default

[ CA_default ]
serial         = $dir/serial
database       = $dir/index.txt
new_certs_dir  = $dir/newcerts
certificate    = $dir/rootCA.crt
private_key    = $dir/private/rootCA.key
default_days   = 365
default_md     = md5
preserve       = no
email_in_dn    = no
nameopt        = default_ca
certopt        = default_ca
policy         = policy_match

[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Чтобы подписать запрос, созданный на предыдущем шаге, выполните следующую команду:

```
# openssl ca -out smtp.turbogaz.net.crt -config ./openssl.cnf -infiles
smtp.turbogaz.net.csr
Using configuration from ./openssl.cnf
Enter pass phrase for ./private/rootCA.key:*******
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
organizationName      :PRINTABLE:'Turbogaz'
organizationalUnitName:PRINTABLE:'Turbogaz SMTP Server'
localityName          :PRINTABLE:'Kharkov'
stateOrProvinceName   :PRINTABLE:'Ukraine'
countryName           :PRINTABLE:'UA'
commonName            :PRINTABLE:'smtp.turbogaz.net'
Certificate is to be certified until May  8 16:20:42 2007 GMT (365 days)
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

В результате выполнения этой команды мы получили два файла и обновили нашу БД:

- smtp.turbogaz.net.crt - непосредственно сам сертификат
- newcerts/<серийный номер>.pem - копия сертификата

Если вы будете пользоваться услугами сторонних центров сертификации, таких как VeriSign, VISA, Thawte, Baltimore и т.д. То Вы должны будете отослать запрос на подпись сертификата (smtp.turbogaz.net.csr) в один из центров сертификации. По истечении некоторого времени, как правило, 1-3 недели вам пришлют готовый и подписанный сертификат (smtp.turbogaz.net.crt). Который уже можно будет использовать непосредственно в ПО. Естественно это услуга платная.

Теперь вы можете проверить сертификат с помощью следующей команды

```
# openssl x509 -in smtp.turbogaz.net.crt -noout -text -purpose
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: O=Turbogaz, OU=Turbogaz Security Division/
        emailAddress=security@turbogaz.net, L=Kharkov, ST=Ukraine, C=UA,
        CN=Turbogaz Root CA
        Validity
            Not Before: May  8 16:20:42 2006 GMT
            Not After : May  8 16:20:42 2007 GMT
        Subject: C=UA, ST=Ukraine, O=Turbogaz, OU=Turbogaz SMTP Server,
        CN=smtp.turbogaz.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (4096 bit)
                Modulus (4096 bit):
                    00:e5:18:e3:71:c4:04:d0:95:55:dc:0b:f6:b9:dd:
                    13:fb:1b:84:93:c1:98:a9:cc:02:ed:6f:dc:b2:94:
                    98:e0:f0:8d:42:80:71:df:d7:47:d6:ad:6b:34:e1:
                    e9:37:3d:75:a0:f2:a1:4f:92:15:9b:04:a8:5e:58:
                    3b:5d:72:e7:36:b0:07:e6:67:6e:57:6f:c6:39:82:
                    91:b9:dc:ec:62:49:d3:90:a5:2f:e5:00:58:5e:49:
                    7d:7f:5d:35:02:5c:15:be:1e:ec:d8:d8:02:ef:ac:
                    8c:7d:c0:a0:47:08:4a:c2:69:31:b9:6f:91:a4:9a:
                    3c:79:7b:da:fc:ea:19:61:bc:bf:ed:52:c8:10:69:
                    60:aa:4a:33:38:31:69:0c:8e:95:f6:c5:3b:cd:58:
                    4e:ba:e5:7a:1a:e4:2e:fd:bd:9f:73:fe:24:6d:1c:
                    88:1a:02:d3:d8:81:1b:00:9a:da:02:2f:4f:38:2d:
                    76:10:d8:36:05:f0:20:13:3c:41:d0:4b:e9:a2:b6:
                    5d:5d:f2:b2:fb:16:bd:41:44:c3:67:54:1f:f3:5c:
                    0b:56:51:45:5e:f9:39:86:50:5c:6c:25:89:e2:cc:
                    e1:0c:42:a8:79:1e:b8:50:f3:99:a1:8b:46:e7:40:
```

```
                    c2:4e:e5:ed:11:4e:61:ff:55:ef:47:f5:b7:82:f4:
                    3e:ab:bb:55:35:1e:11:f5:91:6e:e5:26:54:d7:36:
                    a6:00:26:31:56:04:0f:f0:bc:32:ae:02:ea:63:1d:
                    9a:a5:fe:55:9d:32:f1:2e:e9:85:a4:72:75:ec:b5:
                    6c:9c:21:74:33:a9:74:cf:bc:9f:50:dc:d6:5d:bf:
                    5e:79:c6:f1:db:29:7b:ed:13:a2:e0:90:a2:51:79:
                    bd:af:94:b1:73:38:11:98:ab:3c:b8:2d:5e:1e:84:
                    38:f7:e0:45:67:e8:09:c9:1f:83:35:7f:45:e3:26:
                    ea:53:bb:43:63:07:d8:89:4e:b3:36:7a:b5:4e:93:
                    d9:26:3f:4f:fc:b1:93:ac:32:5b:0e:41:9d:6c:18:
                    4c:7e:59:3e:87:7a:5d:c9:e0:39:30:42:e2:0b:b9:
                    10:db:31:95:3b:56:87:f3:8f:8e:0f:6a:71:00:f2:
                    66:b3:03:ad:ae:a3:a4:61:6d:e6:50:19:6b:9d:e9:
                    30:92:d3:6b:f6:53:9d:4f:7c:12:cb:f9:19:bf:b1:
                    14:42:d5:15:f8:01:34:40:4e:3c:b1:5e:c4:13:ce:
                    60:91:dc:91:b3:37:c0:83:3d:07:97:86:25:5f:14:
                    71:d9:29:f8:f0:71:bf:5a:cc:12:48:2a:08:d5:a8:
                    2c:d0:b9:07:c2:a4:b7:c8:88:34:61:0d:90:fe:5a:
                    ce:1f:07
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        5e:53:3d:bd:f9:a2:3c:8b:2c:fa:52:c1:02:63:81:cf:72:71:
        85:21:52:89:07:f1:d3:b0:9f:91:8e:f7:49:2c:5c:c9:cf:c2:
        ea:df:45:6f:ab:32:29:4a:1a:70:88:6f:99:14:bb:54:dd:b5:
        50:38:3f:9e:28:e5:cc:65:da:33:89:73:48:9f:19:d8:b8:63:
        93:00:97:4e:74:07:df:f8:7a:31:cd:24:ef:63:c6:08:f5:99:
        a4:c0:7c:9b:bf:b3:35:ae:ce:14:bd:92:86:d7:4d:09:58:91:
        b9:72:dd:96:60:e4:b8:60:bf:97:11:3a:04:33:87:d9:bd:1c:
        33:ea:ea:72:8b:e0:72:c5:7b:6b:26:db:d4:28:ec:5b:e0:d7:
        b1:c7:e1:9a:e6:42:1c:51:2d:4d:10:0c:43:09:3d:8e:4b:72:
        e5:c5:59:03:7a:9b:16:cf:51:d4:26:25:9b:75:ca:85:81:c5:
        b1:41:87:b3:53:dd:49:4b:c1:94:b3:4f:88:19:e6:b7:c9:b1:
        1d:09:ac:f3:ea:f3:eb:96:ba:31:f9:54:a8:e6:b9:ac:99:74:
        79:3b:03:2b:b8:ca:3e:0b:60:69:b8:af:72:94:8e:14:29:bb:
        a5:b4:a6:ec:14:c1:92:a6:82:6e:d7:84:76:87:ab:f9:fd:ca:
        8b:23:a1:1c:92:1a:c2:12:41:e3:1a:8b:61:0a:28:33:4d:51:
        b9:a1:41:cf:16:e7:02:d5:98:de:07:75:d9:cf:53:4f:0e:4e:
        79:f6:6f:8d:f6:a9:21:71:8b:ae:b9:02:06:f1:d5:4c:ea:78:
        04:4a:25:10:27:e7:1e:67:7b:53:70:f8:f0:7f:5d:2c:55:ee:
        3b:a6:d1:20:96:28:1b:6a:39:61:be:cb:b4:b2:0f:1d:a2:0d:
        1d:df:7b:b6:9b:f8:36:0c:8a:1e:3c:95:41:24:9e:7b:03:a0:
        55:38:26:1e:0a:6e:77:56:f4:17:c1:be:26:cd:1c:9d:b5:de:
        4e:76:1f:59:d4:13:34:71:b2:ec:ea:e3:e3:8c:86:75:82:18:
        de:f7:72:89:07:c6:7a:04:d4:4b:7a:7a:dc:f8:0c:77:1c:35:
        0b:75:e7:e3:39:fe:68:54:da:04:2a:10:5b:13:44:e7:18:91:
        cd:22:98:52:21:28:ef:f9:44:64:f3:9c:0e:b6:bf:2a:62:2b:
        cc:62:03:f9:8d:52:13:74:0b:e5:cd:89:6b:29:1e:d0:1e:1f:
        d8:2c:6b:86:d0:7b:fd:34:09:96:b5:cf:4c:67:87:30:54:b1:
        b7:c1:15:15:bf:f8:8d:ca:df:b3:5c:6e:2f:d4:5c:24:b7:cd:
        09:8a:29:4b:ab:cb:79:f4
Certificate purposes:
```

```
SSL client : Yes
SSL client CA : No
SSL server : Yes
SSL server CA : No
Netscape SSL server : Yes
Netscape SSL server CA : No
S/MIME signing : Yes
S/MIME signing CA : No
S/MIME encryption : Yes
S/MIME encryption CA : No
CRL signing : Yes
CRL signing CA : No
Any Purpose : Yes
Any Purpose CA : Yes
OCSP helper : Yes
OCSP helper CA : No
```

На данный момент сертификат содержит в себе две версии - закодированную и «удобную для восприятия».

Полезные команды

Проверка корректности закрытого ключа

```
# openssl rsa -check -in test.key
RSA key ok
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAswGJA98yFbc4rQ6maWiaC8aJHHgKe4F+uhTADXOahtRdXjCC
...
mGAvUMZI1GDrmRnI4UVQ+z+sRFY5mNLogH4j2OU5W+VjzXMg10HDMQ==
-----END RSA PRIVATE KEY-----
```

В случае проблем с ключом - вы получите сообщение вида

```
# openssl rsa -check -in test.key
unable to load Private Key
139729410180760:error:0906D066:PEM routines:PEM_read_bio:bad end
line:pem_lib.c:809:

# openssl rsa -check -in test.key
unable to load Private Key
140489218545304:error:0906D064:PEM routines:PEM_read_bio:bad base64
decode:pem_lib.c:823:

# openssl rsa -check -in test.key
unable to load Private Key
139895743596184:error:0D07209B:asn1 encoding routines:ASN1_get_object:too
long:asn1_lib.c:147:
```

```
139895743596184:error:0D068066:asn1 encoding routines:ASN1_CHECK_TLEN:bad
object header:tasn_dec.c:1185:
139895743596184:error:0D07803A:asn1 encoding
routines:ASN1_ITEM_EX_D2I:nested asn1
error:tasn_dec.c:374:Type=PKCS8_PRIV_KEY_INFO
139895743596184:error:0907B00D:PEM routines:PEM_READ_BIO_PRIVATEKEY:ASN1
lib:pem_pkey.c:141:
```

Проверка того, что ssh private и public ключи принадлежат одной паре. Закрытый ключ

```
# openssl rsa -modulus -noout -in id_rsa
Modulus=DD98184A25461E5B9E35A4DD3761F07C296414847B804D5795DEC890E3C20AAA5CED
1D9E10A4BF73018882A4D88CEE479A013303DB542A3770D36345CAF2E10C4F30A6C841A1662D
A286C5FA8373AB866A5286E9F6941925C3BFEE3E6F805162FD1BB25B20382480A3830420223B
AE7E4AD0BBCD2079C0C961C129084F1DEC9E3D8DF108090A2FE3BE240D763AF16AA8C287E55E
48EC454F88350770BB0157E440ABFC86FDC334A74DBA1742F6A751F7FD173251F1458324A48D
D2197D9CA186147F902775A9E85BBE9761F2688AD14B79FC6876F93A604591CF6C45FE85AEBC
3EF675F412E6466D223C1B9DA2C7EB0960B98D0FAF09C2F49CD9C02B78D909E67E475A3279A9
05D7A55F09AC8CE7725DA1D2E45690C50AEFD9E8297A3B6DDEA3C319CB7EB26AFE3902823F68
85A1139CE9C0504B5B015D52DDDCC427B308792ADF8D00C22BFCF7FF5BF476B1BEAFE41B8D01
767EE906E6314DD83DA8F8BCF43D5CCDFF774F9E42B8CB06A6C746703764944380DDBE66D434
DA3BD408009BEB836475D2D39AE98485533CAA79FF3A4ECD03AAE7543527BD4241201EDDF9BF
5CA66647F148429D8B5F3A64DD6110F6A28071E0BE8E74CC43CB3023C819B06A0D65F2752CA7
3C454DE0C7C7ED1812FDBEBA0E96234CA581323087B2C3107B899D4E2CCC7759001DE6A19BAD
A4C735EBA5595B4321EA3CFF7A70FE1F94C9A22863C7
```

Открытый ключ

```
# ssh-keygen -e -m PKCS8 -f id_rsa.pub | openssl rsa -pubin -modulus -noout
Modulus=DD98184A25461E5B9E35A4DD3761F07C296414847B804D5795DEC890E3C20AAA5CED
1D9E10A4BF73018882A4D88CEE479A013303DB542A3770D36345CAF2E10C4F30A6C841A1662D
A286C5FA8373AB866A5286E9F6941925C3BFEE3E6F805162FD1BB25B20382480A3830420223B
AE7E4AD0BBCD2079C0C961C129084F1DEC9E3D8DF108090A2FE3BE240D763AF16AA8C287E55E
48EC454F88350770BB0157E440ABFC86FDC334A74DBA1742F6A751F7FD173251F1458324A48D
D2197D9CA186147F902775A9E85BBE9761F2688AD14B79FC6876F93A604591CF6C45FE85AEBC
3EF675F412E6466D223C1B9DA2C7EB0960B98D0FAF09C2F49CD9C02B78D909E67E475A3279A9
05D7A55F09AC8CE7725DA1D2E45690C50AEFD9E8297A3B6DDEA3C319CB7EB26AFE3902823F68
85A1139CE9C0504B5B015D52DDDCC427B308792ADF8D00C22BFCF7FF5BF476B1BEAFE41B8D01
767EE906E6314DD83DA8F8BCF43D5CCDFF774F9E42B8CB06A6C746703764944380DDBE66D434
DA3BD408009BEB836475D2D39AE98485533CAA79FF3A4ECD03AAE7543527BD4241201EDDF9BF
5CA66647F148429D8B5F3A64DD6110F6A28071E0BE8E74CC43CB3023C819B06A0D65F2752CA7
3C454DE0C7C7ED1812FDBEBA0E96234CA581323087B2C3107B899D4E2CCC7759001DE6A19BAD
A4C735EBA5595B4321EA3CFF7A70FE1F94C9A22863C7
```

Конвертация в формат pfx, который используется на IIS

```
# openssl pkcs12 -export -out domain.name.pfx -inkey domain.name.key -in
domain.name.crt
```

Проверка SNI

```
# openssl s_client -connect site1.example.net:443 -servername
site1.example.net
```

Проверка того, что CSR и сертификат подписаные одним и тем же приватным ключем

```
# openssl req -noout -modulus -in example.net.csr | openssl md5
(stdin)= 496d0967f7fab3d64e3ef9307f27046f

# openssl rsa -noout -modulus -in example.net.key | openssl md5
(stdin)= 496d0967f7fab3d64e3ef9307f27046f

# openssl x509 -noout -modulus -in example.net.crt | openssl md5
(stdin)= 496d0967f7fab3d64e3ef9307f27046f
```

Все md5 должны совпадать.

Для того, чтобы получить rootca с определенного сайта, необходимо выполнить

```
# openssl s_client -connect www.domain.com:443 | tee
www.domain.com-rootca.txt
# openssl x509 -inform PEM -in www.domain.com-rootca.txt -text -out
www.domain.com-rootca.crt
```

Вы можете вырезать «удобную для восприятия» часть из сертификата, полученного на предыдущем шаге с помощью следующей команды:

```
# openssl x509 -in smtp.example.net.crt -out smtp.example.net.crt
```

Для того, чтобы убрать пароль с закрытого ключа (private key) выполните следующую команду. Например, это может понадобится при создании сертификата для postfix, т.к. в документации написано - «The private key must not be encrypted, meaning: the key must be accessible without password».

```
# openssl rsa -in server.key -out nopass.key
writing RSA key
```

Для того, чтобы защитить секретный ключ паролем выполните следующую команду

```
# openssl rsa -des3 -in nopass.key -out pass.key
```

Если вы после этого посмотрите сам ключ в любом текстовом редакторе, то в самом начале ключа должны будут быть подобные строки:

```
# head -3 pass.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,E4786E5109C67B86
```

Данные строки свидетельствуют о том, что данный ключ действительно зашифрован с помощью алгоритма DES3 (TripleDES).

Преобразование сертификата из DER формата в PEM

```
# openssl x509 -inform DER -in smtp.example.net.crt -outform PEM -out
smtp.example.net.pem
```

Преобразование сертификата из PEM формата в DER

```
# openssl x509 -inform PEM -in smtp.example.net.pem -outform DER -out
smtp.example.net.crt
```

Преобразование сертификата из формата PKS#10 (формат, используемый по умолчанию в openssl) в PKCS#12 (формат, используемый по умолчанию Microsoft).

```
# openssl pkcs12 -export -inkey smtp.example.net.key -in
smtp.example.net.crt -out smtp.example.net.p12 -name "Petrov Ivan. Client
certificate."
Enter Export Password: *******
Verifying - Enter Export Password: *******
```

Преобразование сертификата из формата PKS#12 в PEM.

```
# openssl pkcs12 -in smtp.example.net.p12 -out smtp.example.net.pem
Enter Import Password: *******
MAC verified OK
Enter PEM pass phrase: *******
Verifying - Enter PEM pass phrase:  *******
```

Генерация CSR с SAN (Subject Alternative Names)

В файле openssl.cnf в секцию **v3_req** добавляем следующую строку

```
subjectAltName = @alt_names
```

И описываем сами SAN в конце файла

```
[alt_names]
DNS.1 = site-alt-name1.example.net
DNS.2 = site-alt-name2.example.net
DNS.3 = site-alt-name3.example.net
```

Генерируем CSR и приватный ключ

```
# openssl req -nodes -sha256 -newkey rsa:4096 -keyout example.net.key -out
example.net.csr -config ./openssl-san.cnf
```

Проверка SSL сессии с удаленным SMTP сервером с использованием STARTTLS

```
# openssl s_client -starttls smtp -crlf -connect smtp.domain.com:25
```

Проверка SSL сессии с удаленным SMTP сервером

```
# openssl s_client -connect smtp.domain.com:465
```

При использовании команд openssl s_client избегайте использования символа R вначале строки, иначе будете получать сообщение об **RENEGOTIATING**

```
250 DSN
helo www.example.net
250 mail.example.net
mail from:<apache@example.net>
250 2.1.0 Ok
RCPT TO:<root@example.net>
RENEGOTIATING
depth=2 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing,
CN = StartCom Certification Authority
verify return:1
depth=1 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing,
CN = StartCom Class 1 Primary Intermediate Server CA
verify return:1
depth=0 description = G4V86y34KxXe0qbQ, C = US, CN = mail.example.net,
emailAddress = webmaster@example.net
verify return:1
rcpt to:<root@example.net>
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Hello world
.
250 2.0.0 Ok: queued as 4A52616021F
quit
221 2.0.0 Bye
closed
```

Либо используйте опцию **-quiet** для избежания подобного поведения

```
# openssl s_client -starttls smtp -crlf -quiet -connect mail.example.net:25
depth=2 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing,
CN = StartCom Certification Authority
verify return:1
depth=1 C = IL, O = StartCom Ltd., OU = Secure Digital Certificate Signing,
CN = StartCom Class 1 Primary Intermediate Server CA
verify return:1
depth=0 description = G4V86y34KxXe0qbQ, C = US, CN = mail.example.net,
emailAddress = webmaster@example.net
verify return:1
250 DSN
helo www.example.net
250 mail.example.net
mail from:<>
250 2.1.0 Ok
RCPT TO:<root@example.net>
```

```
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
Hellow world 2
.
250 2.0.0 Ok: queued as 65B4616021F
quit
221 2.0.0 Bye
```

Литература

[1] openssl
[2] ssl-cert-howto
[3] Vsevolod A. Stakhov/СЕВКА. "Основы работы с OpenSSL".
[4] Paul Albitz and Cricket Liu. DNS and BIND. O'Reilly. Fourth edition.

openssl.cnf

openssl.cnf

```
# Задаем рабочую директорию
dir      = .

[ ca ]
default_ca    = CA_default

[ CA_default ]
serial         = $dir/serial
database       = $dir/index.txt
new_certs_dir  = $dir/newcerts
certificate    = $dir/rootCA.crt
private_key    = $dir/private/rootCA.key
default_days   = 365
default_md     = md5
preserve       = no
email_in_dn    = no
nameopt        = default_ca
certopt        = default_ca
policy         = policy_match

[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

```
# В данной секции описываются основные опции
[ req ]

# Длина ключа в битах
default_bits = 4096

# Алгоритм шифрования
default_md = md5

# Разрешенные символы
string_mask = nombstr

# Указываем, что DN (Distinguished Name) будет описана в секции
req_distinguished_name
distinguished_name = req_distinguished_name

req_extensions    = v3_req

# В данной секции указываются данные, которые будут использоваться
# по умолчанию при генерации запроса на подписание сертификата
[ req_distinguished_name ]
0.organizationName       = Organization Name (company)
organizationalUnitName   = Organizational Unit Name (department,
division)
emailAddress             = Email Address
emailAddress_max         = 40
localityName             = Locality Name (city, district)
stateOrProvinceName      = State or Province Name (full name)
countryName              = Country Name (2 letter code)
countryName_min          = 2
countryName_max          = 2
commonName               = Common Name (hostname, IP, or your name)
commonName_max           = 64

# Значения по умолчанию
0.organizationName_default    = Turbogaz
localityName_default          = Kharkov
stateOrProvinceName_default   = Ukraine
countryName_default           = UA

[ v3_ca ]
basicConstraints         = CA:TRUE
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid:always,issuer:always

[ v3_req ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
```