Предоставлено: gnutls-bin_3.5.18-1ubuntu1.6_amd64

NAME

certtool - GnuTLS certificate tool

SYNOPSIS

certtool [**-flags**] [**-flag** [значение]] [**--option-name**[[=|]значение]]

Все аргументы должны быть параметрами.

Описание

Инструмент для анализа и генерации сертификатов, запросов и закрытых ключей X.509. Его можно и
интерактивно или неинтерактивно, указав параметр командной строки шаблона.

Инструмент принимает файлы или URL-адреса, поддерживаемые GnuTLS. В случае, если для URL-адрес

доступ вы можете предоставить, используя переменные среды GNUTLS_PIN и GNUTLS_SO_PIN.

Опции

номер -d, **--debug**=номер
Включите отладку. Этот параметр принимает целое число в качестве аргумента. Значение number ограничено значением:
в диапазоне от 0 до 9999

Указывает уровень отладки.

-V, --verbose
Более подробный вывод. Эта опция может отображаться неограниченное количество раз.

--infile=файл
Входной файл.

--outfile=строка
Выходной файл.

-s, --generate-самозаверяющий
Создайте самозаверяющий сертификат.

-c, --generate-certificate
Сгенерируйте подписанный сертификат.

--generate-proxy
Сгенерирует сертификат прокси.

--generate-crl
Сгенерируйте CRL.

Этот параметр генерирует CRL. В сочетании с --load-crl будет использоваться загруженный CRL как база для сгенерированных (т. е. Все отозванные сертификаты в базе будут скопированы в новый CRL).

-u, --update-certificate
Обновите подписанный сертификат.

-p, --generate-privkey
Сгенерируйте закрытый ключ.

--доказуемо

Сгенерируйте закрытый ключ или параметры из исходного кода, используя доказуемый метод.

Для доказуемого ключа будут использоваться алгоритмы FIPS-186-4 (т.Е. Shawe-Taylor) генерация. При указании закрытые ключи или параметры будут сгенерированы из начального значения и могут быть позже проверены с помощью --verify-provable-privkey на правил сгенерированы из начального значения. Вы можете указать --seed или разрешить GnuTLS генерирова (рекомендуется). Этот параметр можно комбинировать с --generate-privkey или --generate-dh-params.

Эта опция применяется к ключам RSA и DSA. Для ключей DSA параметры PQG генерируются с использованием начального значения, а для RSA - двух простых чисел.

--verify-provable-privkey
Проверьте закрытый ключ, сгенерированный из начального файла, используя доказуемый метод.

В нем будут использоваться алгоритмы FIPS-186-4 для доказуемой генерации ключей. Вы можете указать --seed или использовать начальное значение, хранящееся в структуре закрытого ключа.

--seed=строка
При генерации закрытого ключа используйте заданный начальный код в шестнадцатеричном формате.

-q, --generate-request
Создайте запрос сертификата PKCS # 10. Этот параметр не должен отображаться в сочетании ни с одним из следующих параметров: infile.

Сгенерирует запрос сертификата PKCS # 10. Чтобы указать закрытый ключ, используйте --load-privkey.

-e, --verify-chain
Проверьте цепочку сертификатов в кодировке PEM.

Последний сертификат в цепочке должен быть самоподписанным. Его можно комбинировать с --verify-purpose или --verify-hostname .

--verify
Проверка цепочки сертификатов в формате PEM, используя доверенный список.

Список доверенных сертификатов могут быть загружены с-груз-ка-сертификат. Если нет список сертификате указан, то используется список сертификатов системы. Обратите внимание, что во время проверки может быть исследовано несколько путей. При успешной проверке успешный путь будет последним. Его можно комбинировать с --verify-purpose или --verify-hostname .

--verify-crl
Проверьте CRL, используя список доверенных. Этот параметр должен отображаться в сочетании со с : загрузить-са-сертификат.

Список доверенных сертификатов должен быть загружен с помощью --load-ca-certificate .

--verify-hostname=string
Укажите имя хоста, которое будет использоваться для проверки цепочки сертификатов.

Это должно быть объединено с одним из параметров проверки сертификата.

--verify-email=строка
Укажите адрес электронной почты, который будет использоваться для проверки цепочки сертификато
отображаться в сочетании ни с одним из следующих параметров: verify-hostname.

Это должно сочетаться с одним из параметров проверки сертификата.

--verify-purpose=string
Укажите целевой OID, который будет использоваться для проверки цепочки сертификатов.

Этот идентификатор объекта ограничивает назначение сертификатов, подлежащих проверке.
Примерами являются 1.3.6.1.5.5.7.3.1 (TLS WWW), 1.3.6.1.5.5.7.3.4 (ЭЛЕКТРОННАЯ ПОЧТА) и т.д.
Обратите внимание, что сертификат CA без установленного назначения (расширенное использование
любых целей.

--verify-allow-broken
Допускайте для проверки неработающие алгоритмы, такие как MD5.

Это можно комбинировать с --p7-verify, --verify или --verify-chain .

--generate-dh-params
Сгенерируйте параметры Диффи-Хеллмана в кодировке PKCS # 3.

Сгенерирует случайные параметры для использования при обмене ключами Диффи-Хеллмана.
Выходные параметры будут в формате PKCS # 3. Обратите внимание, что рекомендуется использовать
вместо этого параметр --get-dh-params .

--get-dh-params
Получаем включенные параметры Диффи-Хеллмана в кодировке PKCS # 3.

Возвращает сохраненные параметры DH в GnuTLS. Возвращаемые параметры определены в
RFC7919 и могут считаться стандартными параметрами для обмена ключами TLS.

--dh-info
Распечатайте информацию в PKCS #3 с параметрами Диффи-Хеллмана в кодировке.

--load-privkey=string
Загружает файл закрытого ключа.

Это может быть либо файл, либо URL-адрес PKCS # 11

--load-pubkey=строка
Загружает файл с открытым ключом.

Это может быть либо файл, либо URL-адрес PKCS # 11

--load-request=строка
Загружает файл запроса сертификата.

Эту опцию можно использовать с файлом

--load-certificate=string
Загружает файл сертификата.

Эту опцию можно использовать с файлом

--load-ca-privkey=string
Загружает файл закрытого ключа центра сертификации.

Это может быть либо файл, либо URL-адрес PKCS # 11

--load-ca-certificate=строка
Загружает файл сертификата центра сертификации.

Эта опция может использоваться с файлом

--load-crl=string
Загружает предоставленный CRL.

Эта опция может использоваться с файлом

--load-data=string
Загружает вспомогательные данные.

Эту опцию можно использовать с файлом

--password=строка
Используемый пароль.

Вы можете использовать эту опцию, чтобы указать пароль в командной строке вместо считывания его из tty. Обратите внимание, что аргументы командной строки доступны для просмотра в других системах. Указание пароля как " совпадает с указанием пароля нет.

--null-password

Принудительно вводите нулевой пароль.

Этот параметр принудительно вводит нулевой пароль. Это отличается от пустого пароля или его от
пароль в схемах, таких как PKCS # 8.

--empty-password
Принудительно вводите пустой пароль.

Эта опция принудительно вводит пустой пароль. Это отличается от NULL или no
пароль в схемах, таких как PKCS # 8.

--шестнадцатеричные числа
Выведите большое число в более удобном для анализа формате.

--cprint
При определенных операциях он выводит информацию в формате, удобном для C.

При определенных операциях он печатает информацию в формате, удобном для C, подходящем для
включения в программы на C.

-i, --certificate-info
Распечатайте информацию о данном сертификате.

--fingerprint
Распечатайте fingerprint данного сертификата.

Это простой хэш кодировки DER сертификата. Его можно комбинировать
с параметром --hash . Однако для идентификации рекомендуется использовать
key-id, который зависит только от ключа сертификата.

--key-id
Выведите идентификатор ключа данного сертификата.

Это хэш открытого ключа данного сертификата. Он идентифицирует ключ
уникально, остается неизменным при обновлении сертификата и зависит только от подписанных
полей сертификата.

--certificate-pubkey
Распечатайте открытый ключ сертификата.

--pgp-certificate-info
Распечатайте информацию о данном сертификате OpenPGP.

--pgp-ring-info
Распечатайте информацию о заданной структуре набора ключей OpenPGP.

-l, --crl-info
Распечатайте информацию о заданной структуре CRL.

--crq-info
Распечатайте информацию о заданном запросе сертификата.

--no-crq-extensions
Не используйте расширения в запросах сертификатов.

--p12-info
Распечатайте информацию о структуре PKCS # 12.

Эта опция выведет содержимое и распечатает метаданные предоставленной PKCS # 12 структуры.

--p12-name=string
Удобное для использования имя PKCS #12.

Имя, которое будет использоваться для первичного сертификата и закрытого ключа в файле PKCS #

--p7-generate
Создайте структуру PKCS # 7.

Этот параметр создает структуру контейнера сертификатов PKCS # 7. Чтобы добавить сертификаты в структуру, используйте --load-certificate и --load-crl.

--p7-sign
Подписывает, используя структуру PKCS #7.

Эта опция генерирует структуру PKCS # 7, содержащую подпись для предоставленных данных данные из infile. Данные хранятся внутри структуры. Сертификат подписывающего лица должен быть указан с помощью --load-certificate и --load-privkey . Входными данными для --load-certificate может быть список сертификатов. В случае списка первый сертификат используется для подписи, а остальные сертификаты включены в структуру.

--p7-отделенный знак
Подписывает с использованием отдельной структуры PKCS # 7.

Этот параметр генерирует структуру PKCS # 7, содержащую подпись для предоставленных данных из infile. Сертификат подписывающего лица должен быть указан с помощью --load-certificate и --load-privkey . Входными данными для --load-certificate может быть список сертификатов. В случае списка, первый сертификат используется для подписи и другие сертификаты включены в структуру.

--p7-include-cert, --no-p7-include-cert

Сертификат подписавшего будет включен в список сертификатов.. Форма no-p7-include-cert отключит эту опцию. Эта опция включена по умолчанию.

Эти опции работают с --p7-sign или --p7-detached-sign и будут включать или исключать сертификат подписывающего лица в сгенерированную подпись.

--p7-time, --no-p7-time
Будет включать временную метку в структуру PKCS # 7. На нет-P7-то время форма будет отключить опцию.

Этот вариант будет включать в себя временную метку в созданном подпись

--p7-показать-данные, --нет-p7-показать-данные
Отобразит встроенные данные в структуре PKCS # 7. Форма no-p7-show-data отключит эту опцию.

Этот параметр можно комбинировать с --p7-verify или --p7-info, и он отобразит встроенные подписанные данные в структуре PKCS #7.

--p7-info
Распечатайте информацию о структуре PKCS # 7.

--p7-verify
Проверьте предоставленную структуру PKCS # 7.

Этот параметр проверяет подписанную структуру PKCS # 7. Список сертификатов, который будет исп
проверки, можно указать с помощью --load-ca-certificate . Если список сертификатов не предоста
, используется системный список сертификатов. В качестве альтернативы можно предоставить пряму
с помощью --load-certificate . Назначение ключа может быть реализовано с помощью опции --verify-purpose , а опция --load-data будет использовать отдельные данные.

--p8-info
Печать информации о структуре PKCS # 8.

Эта опция будет печатать информацию о зашифрованных структурах PKCS # 8. Этот параметр не требует расшифровки структуры.

--smime-to-p7
Преобразуйте S / MIME в структуру PKCS # 7.

-k, --key-info
Распечатайте информацию о закрытом ключе.

--pgp-key-info
Распечатайте информацию с помощью закрытого ключа OpenPGP.

--pubkey-info
Распечатайте информацию с помощью открытого ключа.

Опция в сочетании с --load-request, --load-pubkey, --load-privkey и --load-certificate извлекает открытый ключ рассматриваемого объекта.

--v1 Сгенерируйте сертификат X.509 версии 1 (без расширений).

--to-p12
Сгенерируйте структуру PKCS # 12.

Для этого требуется указать сертификат, закрытый ключ и, возможно, сертификат CA
.

--to-p8
Создайте структуру PKCS # 8.

-8, --pkcs8
Используйте формат PKCS # 8 для закрытых ключей.

--rsa Генерирует ключ RSA.

В сочетании с --generate-privkey генерирует закрытый ключ RSA.

--dsa Генерирует ключ DSA.

В сочетании с --generate-privkey генерирует закрытый ключ DSA.

--ecc Генерирует ключ ECC (ECDSA).

В сочетании с --generate-privkey генерирует закрытый ключ эллиптической кривой, который будет использоваться с ECDSA.

--ecdsa
Это псевдоним для опции --ecc.

--hash=строка
Алгоритм хэширования, используемый для подписи.

Доступны следующие хэш-функции: SHA1, RMD160, SHA256, SHA384, SHA512, SHA3-224, SHA3-256, SHA3-384, SHA3-512.

--inder, --no-inder
Используйте формат DER для ввода сертификатов, закрытых ключей и параметров DH. В форме без индексации эта опция будет отключена.

Предполагается, что входные файлы будут в формате DER или RAW. В отличие от опций, которые в PEM input допускают несколько входных данных (например, несколько сертификатов), при чтении в формате DER считывается одна структура данных.

--inraw
Это псевдоним для опции --inder.

--outder, --no-outder
Используйте формат DER для вывода сертификатов, закрытых ключей и параметров DH. Форма без отправки отключит эту опцию.

Выходные данные будут в формате DER или RAW.

--outraw
Это псевдоним для опции --outder.

--bits=число
Укажите количество битов для генерации ключа. Этот параметр принимает целое число в качестве аргумента.

--curve=строка
Укажите кривую, используемую для генерации ключа EC.

Поддерживаемые значения secp192r1, secp224r1, secp256r1, secp384r1 и secp521r1.

--sec-param=параметр безопасности.
Укажите уровень безопасности [низкий, устаревший, средний, высокий, ultra].

Это альтернатива опции bits.

--disable-quick-random
Никакого эффекта.

--template=string
Файл шаблона для использования в неинтерактивной работе.

--stdout-info
Распечатайте информацию в stdout вместо stderr.

--ask-pass
Включите взаимодействие для ввода пароля в пакетном режиме..

Эта опция позволит взаимодействовать для ввода пароля в пакетном режиме. Это полезно, когда указан параметр шаблона.

--pkcs-cipher=шифр
Шифр, используемый для операций PKCS # 8 и # 12.

Шифр может быть одним из 3des, 3des-pkcs12, aes-128, aes-192, aes-256, rc2-40, arcfour.

--provider=string
Укажите библиотеку поставщика PKCS #11.

Это переопределит параметры по умолчанию в /etc/gnutls/pkcs11.conf

-h, --help
Отобразите информацию об использовании и завершите работу.

-!, --подробнее-справка
Передайте расширенную информацию об использовании через пейджер.

-v [{v | c | n **--version** [{v | c | n}]}]
Выведите версию программы и завершите работу. Режим по умолчанию - "v", простая версия.
В режиме "c" будет напечатана информация об авторских правах, а "n" - полное уведомление об ав
.

FILES

Certtool's template file format
A template file can be used to avoid the interactive questions of certtool. Initially create a file named 'cert.cfg' that contains the information about the certificate. The template can be used as below:

```
$ certtool --generate-certificate --load-privkey key.pem     --template cert.cfg --c
```

An example certtool template file that can be used to generate a certificate request or self signed certificate follows.

```
# X.509 Certificate options
#
# DN options

# The organization of the subject.
organization = "Koko inc."

# The organizational unit of the subject.
unit = "sleeping dept."
```

```
# The locality of the subject.
# locality =

# The state of the certificate owner.
state = "Attiki"

# The country of the subject. Two letter code.
country = GR

# The common name of the certificate owner.
cn = "Cindy Lauper"

# A user id of the certificate owner.
#uid = "clauper"

# Set domain components
#dc = "name"
#dc = "domain"

# If the supported DN OIDs are not adequate you can set
# any OID here.
# For example set the X.520 Title and the X.520 Pseudonym
# by using OID and string pairs.
#dn_oid = "2.5.4.12 Dr."
#dn_oid = "2.5.4.65 jackal"

# This is deprecated and should not be used in new
# certificates.
# pkcs9_email = "none@none.org"

# An alternative way to set the certificate's distinguished name directly
# is with the "dn" option. The attribute names allowed are:
# C (country), street, O (organization), OU (unit), title, CN (common name),
# L (locality), ST (state), placeOfBirth, gender, countryOfCitizenship,
# countryOfResidence, serialNumber, telephoneNumber, surName, initials,
# generationQualifier, givenName, pseudonym, dnQualifier, postalCode, name,
# businessCategory, DC, UID, jurisdictionOfIncorporationLocalityName,
# jurisdictionOfIncorporationStateOrProvinceName,
# jurisdictionOfIncorporationCountryName, XmppAddr, and numeric OIDs.

#dn = "cn = Nikos,st = New Something,C=GR,surName=Mavrogiannopoulos,2.5.4.9=Arkadias

# The serial number of the certificate
# Comment the field for a time-based serial number.
serial = 007
```

```
# In how many days, counting from today, this certificate will expire.
# Use -1 if there is no expiration date.
expiration_days = 700

# Alternatively you may set concrete dates and time. The GNU date string
# formats are accepted. See:
# http://www.gnu.org/software/tar/manual/html_node/Date-input-formats.html

#activation_date = "2004-02-29 16:21:42"
#expiration_date = "2025-02-29 16:24:41"

# X.509 v3 extensions

# A dnsname in case of a WWW server.
#dns_name = "www.none.org"
#dns_name = "www.morethanone.org"

# An othername defined by an OID and a hex encoded string
#other_name = "1.3.6.1.5.2.2 302ca00d1b0b56414e5245494e2e4f5247a11b3019a006020400000
#other_name_utf8 = "1.2.4.5.6 A UTF8 string"
#other_name_octet = "1.2.4.5.6 A string that will be encoded as ASN.1 octet string"

# Allows writing an XmppAddr Identifier
#xmpp_name = juliet@im.example.com

# Names used in PKINIT
#krb5_principal = user@REALM.COM
#krb5_principal = HTTP/user@REALM.COM

# A subject alternative name URI
#uri = "http://www.example.com"

# An IP address in case of a server.
#ip_address = "192.168.1.1"

# An email in case of a person
email = "none@none.org"

# TLS feature (rfc7633) extension. That can is used to indicate mandatory TLS
# extension features to be provided by the server. In practice this is used
# to require the Status Request (extid: 5) extension from the server. That is,
# to require the server holding this certificate to provide a stapled OCSP response.
# You can have multiple lines for multiple TLS features.

# To ask for OCSP status request use:
#tls_feature = 5
```

```
# Challenge password used in certificate requests
challenge_password = 123456

# Password when encrypting a private key
#password = secret

# An URL that has CRLs (certificate revocation lists)
# available. Needed in CA certificates.
#crl_dist_points = "http://www.getcrl.crl/getcrl/"

# Whether this is a CA certificate or not
#ca

# Subject Unique ID (in hex)
#subject_unique_id = 00153224

# Issuer Unique ID (in hex)
#issuer_unique_id = 00153225

#### Key usage

# The following key usage flags are used by CAs and end certificates

# Whether this certificate will be used to sign data (needed
# in TLS DHE ciphersuites). This is the digitalSignature flag
# in RFC5280 terminology.
signing_key

# Whether this certificate will be used to encrypt data (needed
# in TLS RSA ciphersuites). Note that it is preferred to use different
# keys for encryption and signing. This is the keyEncipherment flag
# in RFC5280 terminology.
encryption_key

# Whether this key will be used to sign other certificates. The
# keyCertSign flag in RFC5280 terminology.
#cert_signing_key

# Whether this key will be used to sign CRLs. The
# cRLSign flag in RFC5280 terminology.
#crl_signing_key

# The keyAgreement flag of RFC5280. It's purpose is loosely
# defined. Not use it unless required by a protocol.
#key_agreement
```

```
# The dataEncipherment flag of RFC5280. It's purpose is loosely
# defined. Not use it unless required by a protocol.
#data_encipherment

# The nonRepudiation flag of RFC5280. It's purpose is loosely
# defined. Not use it unless required by a protocol.
#non_repudiation

#### Extended key usage (key purposes)

# The following extensions are used in an end certificate
# to clarify its purpose. Some CAs also use it to indicate
# the types of certificates they are purposed to sign.

# Whether this certificate will be used for a TLS client;
# this sets the id-kp-serverAuth (1.3.6.1.5.5.7.3.1) of
# extended key usage.
#tls_www_client

# Whether this certificate will be used for a TLS server;
# This sets the id-kp-clientAuth (1.3.6.1.5.5.7.3.2) of
# extended key usage.
#tls_www_server

# Whether this key will be used to sign code. This sets the
# id-kp-codeSigning (1.3.6.1.5.5.7.3.3) of extended key usage
# extension.
#code_signing_key

# Whether this key will be used to sign OCSP data. This sets the
# id-kp-OCSPSigning (1.3.6.1.5.5.7.3.9) of extended key usage extension.
#ocsp_signing_key

# Whether this key will be used for time stamping. This sets the
# id-kp-timeStamping (1.3.6.1.5.5.7.3.8) of extended key usage extension.
#time_stamping_key

# Whether this key will be used for email protection. This sets the
# id-kp-emailProtection (1.3.6.1.5.5.7.3.4) of extended key usage extension.
#email_protection_key

# Whether this key will be used for IPsec IKE operations (1.3.6.1.5.5.7.3.17).
#ipsec_ike_key

## adding custom key purpose OIDs
```

```
# for microsoft smart card logon
# key_purpose_oid = 1.3.6.1.4.1.311.20.2.2

# for email protection
# key_purpose_oid = 1.3.6.1.5.5.7.3.4

# for any purpose (must not be used in intermediate CA certificates)
# key_purpose_oid = 2.5.29.37.0

### end of key purpose OIDs

### Adding arbitrary extensions
# This requires to provide the extension OIDs, as well as the extension data in
# hex format. The following two options are available since GnuTLS 3.5.3.
#add_extension = "1.2.3.4 0x0AAB01ACFE"

# As above but encode the data as an octet string
#add_extension = "1.2.3.4 octet_string(0x0AAB01ACFE)"

# For portability critical extensions shouldn't be set to certificates.
#add_critical_extension = "5.6.7.8 0x1AAB01ACFE"

# When generating a certificate from a certificate
# request, then honor the extensions stored in the request
# and store them in the real certificate.
#honor_crq_extensions

# Alternatively only specific extensions can be copied.
#honor_crq_ext = 2.5.29.17
#honor_crq_ext = 2.5.29.15

# Path length contraint. Sets the maximum number of
# certificates that can be used to certify this certificate.
# (i.e. the certificate chain length)
#path_len = -1
#path_len = 2

# OCSP URI
# ocsp_uri = http://my.ocsp.server/ocsp

# CA issuers URI
# ca_issuers_uri = http://my.ca.issuer

# Certificate policies
#policy1 = 1.3.6.1.4.1.5484.1.10.99.1.0
```

```
#policy1_txt = "This is a long policy to summarize"
#policy1_url = http://www.example.com/a-policy-to-read

#policy2 = 1.3.6.1.4.1.5484.1.10.99.1.1
#policy2_txt = "This is a short policy"
#policy2_url = http://www.example.com/another-policy-to-read

# Name constraints

# DNS
#nc_permit_dns = example.com
#nc_exclude_dns = test.example.com

# EMAIL
#nc_permit_email = "nmav@ex.net"

# Exclude subdomains of example.com
#nc_exclude_email = .example.com

# Exclude all e-mail addresses of example.com
#nc_exclude_email = example.com

# IP
#nc_permit_ip = 192.168.0.0/16
#nc_exclude_ip = 192.168.5.0/24
#nc_permit_ip = fc0a:eef2:e7e7:a56e::/64

# Options for proxy certificates
#proxy_policy_language = 1.3.6.1.5.5.7.21.1

# Options for generating a CRL

# The number of days the next CRL update will be due.
# next CRL update will be in 43 days
#crl_next_update = 43

# this is the 5th CRL by this CA
# Comment the field for a time-based number.
#crl_number = 5

# Specify the update dates more precisely.
#crl_this_update_date = "2004-02-29 16:21:42"
#crl_next_update_date = "2025-02-29 16:24:41"

# The date that the certificates will be made seen as
# being revoked.
```

```
    #crl_revocation_date = "2025-02-29 16:24:41"
```

EXAMPLES

Generating private keys

To create an RSA private key, run:

```
    $ certtool --generate-privkey --outfile key.pem --rsa
```

To create a DSA or elliptic curves (ECDSA) private key use the above command combined wi
'dsa' or 'ecc' options.

Generating certificate requests

To create a certificate request (needed when the certificate is issued by another party), run:

```
    certtool --generate-request --load-privkey key.pem    --outfile request.pem
```

If the private key is stored in a smart card you can generate a request by specifying th
private key object URL.

```
    $ ./certtool --generate-request --load-privkey "pkcs11:..."   --load-pubkey "pkcs11:
```

Generating a self-signed certificate

To create a self signed certificate, use the command:

```
    $ certtool --generate-privkey --outfile ca-key.pem
    $ certtool --generate-self-signed --load-privkey ca-key.pem    --outfile ca-cert.pem
```

Note that a self-signed certificate usually belongs to a certificate authority, that sig
other certificates.

Generating a certificate

To generate a certificate using the previous request, use the command:

```
    $ certtool --generate-certificate --load-request request.pem    --outfile cert.pem -
```

To generate a certificate using the private key only, use the command:

```
    $ certtool --generate-certificate --load-privkey key.pem    --outfile cert.pem --loa
```

Certificate information

To view the certificate information, use:

```
    $ certtool --certificate-info --infile cert.pem
```

PKCS #12 structure generation

To generate a PKCS #12 structure using the previous key and certificate, use the command

```
    $ certtool --load-certificate cert.pem --load-privkey key.pem    --to-p12 --outder -
```

Some tools (reportedly web browsers) have problems with that file because it does not contain the CA certificate for the certificate. To work around that problem in the tool you can use the --load-ca-certificate parameter as follows:

```
$ certtool --load-ca-certificate ca.pem   --load-certificate cert.pem --load-privkey
```

Diffie-Hellman parameter generation

To generate parameters for Diffie-Hellman key exchange, use the command:

```
$ certtool --generate-dh-params --outfile dh.pem --sec-param medium
```

Proxy certificate generation

Proxy certificate can be used to delegate your credential to a temporary, typically short-lived, certificate. To create one from the previously created certificate, first create a temporary key and then generate a proxy certificate for it, using the commands:

```
$ certtool --generate-privkey > proxy-key.pem
$ certtool --generate-proxy --load-ca-privkey key.pem   --load-privkey proxy-key.pem
```

Certificate revocation list generation

To create an empty Certificate Revocation List (CRL) do:

```
$ certtool --generate-crl --load-ca-privkey x509-ca-key.pem            --load-ca-cer
```

To create a CRL that contains some revoked certificates, place the certificates in a fil and use **--load-certificate** as follows:

```
$ certtool --generate-crl --load-ca-privkey x509-ca-key.pem   --load-ca-certificate
```

To verify a Certificate Revocation List (CRL) do:

```
$ certtool --verify-crl --load-ca-certificate x509-ca.pem < crl.pem
```

EXIT STATUS

One of the following exit values will be returned:

0 (EXIT_SUCCESS)
Successful program execution.

1 (EXIT_FAILURE)
The operation failed or the command syntax was not valid.

70 (EX_SOFTWARE)

libopts had an internal operational error. Please report it to autogen-users@lists.sourceforge.net. Thank you.

SEE ALSO

p11tool (1)

AUTHORS

Nikos Mavrogiannopoulos, Simon Josefsson and others; see /usr/share/doc/gnutls/AUTHORS f
a complete list.

COPYRIGHT

BUGS

Please send bug reports to: bugs@gnutls.org

NOTES

This manual page was AutoGen-erated from the **certtool** option definitions.