КРИПТОКОМ

УТВЕРЖДЕН
СЕИУ.00009-04 34 03 - ЛУ

Подп. и дата
Инв. № дубл.
Взам. инв.№
Подп. и дата
Инв.№ подп.

СРЕДСТВО КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ
«МагПро КриптоПакет» 3.0

Утилита openssl

Руководство по использованию

СЕИУ.00009-04 34 03
Листов 143

Литера О

2017

Аннотация

Настоящий документ содержит руководство по использованию утилиты openssl, которая представляет собой исполнение 3 (соответствует классу КС1) и исполнение 4 (соответствует классу КС2) СКЗИ «МагПро КриптоПакет» 3.0.

Авторские права на «МагПро КриптоПакет» 3.0 принадлежат ООО «Криптоком».

В СКЗИ использован код OpenSSL, ©1998-2017 The OpenSSL Project.

«МагПро» является зарегистрированной торговой маркой ООО «Криптоком».

Содержание

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

1 Назначение

Утилита openssl — это командно-строчная утилита для выполнения различных криптографических операций из командной оболочки. С ее помощью можно выполнять криптографические преобразования файлов (в том числе файла стандартного ввода), прежде всего в форматах CMS и S/MIME, а также формировать ключевую информацию, метки доверенного времени и реализовывать протокол онлайн-проверки статуса сертификата.

Также утилита может использоваться как простейшие tls-клиент и tls-сервер.

Утилита openssl — это составная часть СКЗИ «МагПро КриптоПакет» 3.0, а именно исполнение 3 (соответствует классу КС1) и исполнение 4 (соответствует классу КС2) указанного СКЗИ.

Утилита openssl является функционально законченным изделием.

Все криптографические преобразования и протоколы основаны, в первую очередь, на алгоритмах ГОСТ. Алгоритмы шифрования и расшифровывания соответствуют ГОСТ 28147-89, алгоритмы имитозащиты соответствуют ГОСТ 28147-89 и HMAC ГОСТ Р 34.11-94/2012, алгоритмы вычисления хэш-функции соответствуют ГОСТ Р 34.11-94/2012, алгоритмы создания и проверки ЭП соответствуют ГОСТ Р 34.10-2001/2012. В большинстве случаев алгоритмы ГОСТ используются по умолчанию, в некоторых командах использование алгоритмов ГОСТ необходимо задавать специальным ключом. Подробнее об особенности задания криптоалгоритмов см. раздел 7.2

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

2 Условия выполнения программы

Утилита openssl предназначена для работы в следующих операционных системах:

Windows 7 SP1/8.1/10;
Windows Server 2008R2 SP1/2012/2012R2/2016
Debian GNU/Linux 7(wheezy)/8(jessie)/stretch;
Linux Mint 17.x, 18.x, Linux Mint Debian Edition 2
Ubuntu 14.04, 16.04;
RedHat Enterprise Linux 6, 7;
CentOS 6, 7;
SUSE Linux 11, 12;
OpenSUSE 42.2, 42.3;
OS EMIAS 1.0;
Альт Линукс 6, 7, 8;
МСВСфера Сервер 6.3, МСВСфера АРМ 6.3;
Атликс 3.1;
Гослинукс IC4;
FreeBSD 10.x, 11.x;
Oracle Solaris 10, 11;
MacOS 10.12;
Rosa Enterprise Desktop (RED) X2, X3;
Rosa Enterprise Linux Server (RELS) 6, 7; РОСА КОБАЛЬТ 1.0;
Astra Linux Special Edition РУСБ.10015-07.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

3 Перечень функций

«МагПро КриптоПакет» 3.0 в исполнении Утилита openssl реализует следующие функции:

- создание и проверка электронной подписи данных, представленных в виде файлов (в том числе файла стандартного ввода), в соответствии с ГОСТ Р 34.10;
- зашифрование и расшифрование данных, представленных в виде файлов (в том числе файла стандартного ввода), в соответствии с ГОСТ 28147-89;
- имитозащита данных, представленных в виде файлов (в том числе файла стандартного ввода), в соответствии с ГОСТ 28147-89 и HMAC ГОСТ Р 34.11;
- вычисление значения хэш-функции данных, представленных в виде файлов (в том числе файла стандартного ввода), в соответствии с ГОСТ Р 34.11;
- вычисление ключа парной связи по алгоритму VKO с использованием как долговременных, так и эфемерных пар закрытых и открытых ключей, созданных в соответствии с ГОСТ Р 34.10;
- выработка случайного числа заданной длины;
- создание закрытых ключей и ключей электронной подписи в соответствии с ГОСТ Р 34.10;
- вычисление открытых ключей и ключей проверки подписи в соответствии с ГОСТ Р 34.10, формирование запросов на выпуск сертификата;
- формирование и проверка сертификатов открытых ключей и ключей проверки подписи в формате X.509, а также списков отзыва сертификатов;
- экспорт ключей в транспортный контейнер и импорт ключей из транспортного контейнера в формате PKCS#12;
- криптографическая обработка сообщений в форматах CMS и S/MIME;
- реализация протокола получения меток доверенного времени;
- реализация протокола OCSP онлайн-проверки статуса сертификата;
- реализации протокола TLS с использованием российских наборов алгоритмов шифрования TLS_GOSTR341112_256_WITH_28147_CNT_IMIT и TLS_GOSTR341001_WITH_28147_CNT_IMIT (набор TLS_GOSTR341001_WITH_28147_CNT_IMIT следует использовать только для соединения с серверами, не поддерживающими набор TLS_GOSTR341112_256_WITH_28147_CNT_IMIT).

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

4 Обеспечение информационной безопасности при использовании «МагПро КриптоПакет» 3.0

Надежная криптографическая защита данных при использовании «МагПро КриптоПакет» 3.0 обеспечивается только в том случае, если эксплуатация «МагПро КриптоПакет» 3.0 осуществляется в строгом соответствии с требованиями документа «Средство криптографической защиты информации «МагПро КриптоПакет» 3.0. Правила пользования» (СЕИУ.СЕИУ.00009–04 94).

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

5 Установка

5.1 Установка для ОС семейства Windows

Для установки «МагПро КриптоПакет» 3.0 в исполнении Утилита openssl воспользуйтесь инсталлятором «МагПро КриптоПакет» 3.0.

5.2 Установка для ОС, не входящих в семейство Windows

Для установки «МагПро КриптоПакет» 3.0 в исполнении Утилита openssl выполните следующие действия:

1. Перейдите в каталог с установочными файлами (пакетами).
2. С помощью системных утилит (dpkg, apt, rpm, yum, dnf и т.д.) установите пакеты *openssl—r** подходящей архитектуры. Возникающие зависимости следует разрешать с помощью системного или иного репозитория, которому Вы доверяете. Установка производится в каталог */opt/cryptopack3*.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

6 Настройка

6.1 Общие замечания

В OpenSSL приложения могут автоматически конфигурировать некоторые аспекты OpenSSL, используя главный конфигурационный файл OpenSSL или, опционально, альтернативный конфигурационный файл. Утилита openssl включает эту функциональность: любая ее команда использует главный конфигурационный файл OpenSSL, если в команде не указана опция использования альтернативного конфигурационного файла.

6.2 Конфигурационные файлы OpenSSL

6.2.1 Описание

Конфигурационный файл библиотеки OpenSSL разделен на ряд секций. Каждая секция начинается со строки [section_name] и заканчивается там, где начинается следующая секция или заканчивается файл. Имя секции может состоять из букв, цифр и знаков подчеркивания.

Первая секция конфигурационного файла особая. О ней говорят как об умолчательной секции (default section). Обычно она не именована и располагается от начала файла до первой именованной секции. Когда ищут какое-то имя, его сначала ищут в именованных секциях (если таковые есть), а затем в умолчательной секции.

Окружение отображено в секции ENV (т.е. на переменные окружения можно ссылаться как на переменные секции ENV).

Комментарии можно включать, указывая в начале строки знак #.

Каждая секция в конфигурационном файле состоит из некоторого количества пар имя-значение в формате

```
имя=значение
```

Строка «имя» может содержать любые латинские буквы и цифры, а также некоторые знаки пунктуации: . , ; и _.

Строка «значение» состоит из строки, следующей за символом = и продолжающейся до конца строки, но не включает предстоящий и последующий пробелы.

В строке «значение» производятся подстановки переменных. Это можно сделать, включив синтаксическую конструкцию $var или $var: так подставляется значение названной переменной в текущей секции. Можно также подставить значение из другой секции, используя синтаксис $section::name или $section::name. Используя конструкцию $ENV::name, можно подставлять переменные окружения. Можно также присваивать значения переменным окружения, указывая в качестве имени ENV::name, это будет работать, если программа обращается к переменным окружения с помощью библиотеки CONF, а не непосредственно вызывая функцию getenv().

Можно употребить некоторые специальные символы как обычные, если заключить содержащую их строку в кавычки или поставить перед символом знак \. Если в конце строки «значение» поставить \, то можно перенести ее на следующую строку файла. Кроме того, распознаются последовательности знаков \n, \r, \b, \t.

6.2.2 Конфигурация библиотеки OpenSSL

Для того, чтобы обеспечить возможность конфигурирования библиотеки, умолчательная секция должна содержать соответствующую строку, указывающую на главную конфигура-

ционную секцию. Умолчательное имя, используемое утилитой openssl - openssl_conf. Другие приложения могут использовать альтернативное имя, например myapplicaton_conf.

Конфигурационная секция должна состоять из набора пар имя-значение, содержащего модульно-специфичную конфигурационную информацию. Строка «имя» представляет собой имя конфигурационного модуля, значение строки «значение» более специфично: оно может, например, представлять следующую конфигурационную секцию, содержащую информацию, специфичную для конфигурационного модуля. Например:

```
openssl_conf = openssl_init

        [openssl_init]

        oid_section = new_oids
        engines = engine_section

        [new_oids]

        ... new oids here ...

        [engine_section]

        ... engine stuff here ...
```

В настоящее время существуют два конфигурационных модуля. Один - для ASN1-объектов, другой для конфигурирования модулей ENGINE.

6.2.3 Конфигурирование поддержки алгоритмов ГОСТ

Чтобы включить поддержку алгоритмов ГОСТ, необходимо описать в конфигурационном файле подключение модуля engine libcryptocom.

При установке соответствующего пакета автоматически выполняется подключение этого модуля в системном конфигурационном файле.

Для подключения необходимо добавить в конфигурационный файл следующую информацию:

1. До названия первой секции (первая строка [в квадратных скобках]) следует поместить команду `openssl_conf`, указывающую на секцию с глобальными параметрами конфигурации (по умолчанию этой секции не существует в файле, ее необходимо добавить):

   ```
   openssl_conf = openssl_def
   ```

2. Добавить в конфигурационный файл (например, в конец файла) секцию, указанную выше, и вставить в нее команду `engines`, указывающую на секцию со спискоммодулей, которые необходимо подгрузить:

   ```
   [openssl_def]
   engines = engine_section
   ```

3. Добавить в конфигурационный файл секцию `engines`, содержащую строку с ID МагПро Engine и название секции, описывающей его конфигурацию:

   ```
   [engine_section]
   cryptocom = cryptocom_section %(для Cryptocom Engine)
   ```

4. Добавить в конфигурационный файл секцию, описывающую конфигурацию библиотеки. Эта секция должна содержать по меньшей мере две строки — в одной указывается полный путь к модулю, во второй указывается его ID.

```
[cryptocom_section]
engine_id = cryptocom
default_algorithms = ALL
```

Кроме этого, в секцию конфигурации библиотеки libcryptocom может быть включена конфигурационная информация самой библиотеки.

Эта информация включает в себя три параметра:

`RNG` — Тип датчика случайных чисел. Допустимые значения: `PROGRAM`, `ACCORD`, `SOBOL`, `VJUGA`.

`RNG_PARAMS` — Дополнительные параметры датчика случайных чисел. Для программного датчика этот параметр указывает на расположения файла начального заполнения программного ДСЧ, если его местоположение не совпадает с умолчательным.

`CRYPT_PARAMS` — Параметры алгоритма шифрования ГОСТ 28147-89. Значением опции является OID параметров алгоритма шифрования (см табл. 2), который будет использоваться для зашифрования документов. На работу TLS этот параметр не влияет, так как параметры шифрования жестко фиксированы в спецификации шифрсьютов TLS.

Эти параметры конфигурации могут быть также заданы в environment с помощью переменных с теми же именами и значениями. Значения, заданные в environment, имеют приоритет перед значениями в конфигурационном файле.

Некоторые приложения (например, apache/mod_ssl, stunnel, openvpn) не считывают конфигурационный файл libcrypto, а предоставляют собственные средства конфигурации, позволяющие загружать модули engine. При использовании этих приложений необходимо в их конфигурационном файле указать использование engine с идентификатором cryptocom, а параметры библиотеки libcryptocom передавать через environment.

6.2.4 Конфигурационный модуль для ASN1-объектов

Этот модуль имеет имя oid_section. Значение этой переменной указывает на секцию, содержащую пары имя-значение для OIDов: имя — длинное и короткое имя OID, значение - численная форма OID. Хотя некоторые из команд утилиты openssl уже имеют собственную функциональность секции ASN1-объектов, но не все. При использовании конфигурационного модуля для ASN1-объектов все команды утилиты openssl также могут видеть новые объекты, как и все совместимые приложения. Например:

```
[new_oids]

some_new_oid = 1.2.3.4
some_other_oid = 1.2.3.5
```

В OpenSSL 0.9.8 также возможно установить в качестве значения длинное имя, за которым идет запятая и численная форма OID. Например:

```
shortName = some object long name, 1.2.3.4
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

6.2.5 Конфигурационный модуль для модулей ENGINE

Этот конфигурационный модуль для модулей ENGINE (энджин) имеет имя engines. Значение этой переменной указывает на секцию, содержащую дальнейшую конфигурационную информацию для модулей ENGINE.

Секция, на которую указывает engines, является таблицей имен модулей (но см. engine_id ниже) и дальнейших секций, содержащих конфигурационную информацию, специфичную для каждого модуля ENGINE.

Каждая такая ENGINE-специфичная секция используется для установки алгоритмов по умолчанию, загрузки динамически загружаемых модулей, выполнения инициализации и отправки управляющих команд. Какая именно операция выполняется, зависит от имени command, которое является именем в паре имя-значение. Поддерживаемые в настоящее время команды перечислены ниже.

Например:

```
[engine_section]

# Configure ENGINE named "libcryptocom"
libcryptocom = libcrypocom_section
# Configure ENGINE named "bar"
bar = bar_section

[libcryptocom_section]
... libcryptocom ENGINE specific commands ...

[bar_section]
... "bar" ENGINE specific commands ...
```

Команда engine_id используется для задания имени модуля ENGINE. Если эта команда используется, она должна идти первой. Например:

```
[engine_section]

# This would normally handle an ENGINE named "libcryptocom"
libcryptocom = libcryptocom_section

[libcryptocom_section]
# Override default name and use "mylibcryptocom" instead.
engine_id = mylibcryptocom
```

Команда dynamic_path загружает и добавляет модуль ENGINE с указанного пути. Она эквивалента отправке управляющей команды SO_PATH с аргументом path, затем команды LIST_ADD со значением 2 и LOAD динамически загружаемому модулю ENGINE. Если требуется иное поведение, можно отправить альтернативные управляющие команды непосредственно динамически загружаемому модулю ENGINE, использующему управляющие команды.

Команда init определяет, надо ли инициализировать модуль ENGINE. Если ее значение 0, то модуль ENGINE не будет инициализирован, если 1, то делается попытка немедленно

инициализировать модуль ENGINE. Если команда init отсутствует, то будет сделана попытка инициализировать модуль ENGINE после того, как будут отданы все команды в этой секции.

Команда default_algorithms устанавливает алгоритмы по умолчанию, которые будет поддерживать модуль ENGINE, используя функции ENGINE_set_default_string().

Если имя не соответствует ни одной из приведенных выше названий команд, предполагается, что это управляющая команда, которая отправляется модулю ENGINE. Значение этой строки в конфигурационном файле — аргумент этой управляющей команды. Если значение - строка EMPTY, то никакого аргумента команде не передается.

Например:

```
[engine_section]

# Configure ENGINE named "libcryptocom"
libcryptocom = libcryptocom_section

[libcryptocom_section]
# Load engine from DSO

dynamic_path = /some/path/libcryptocomengine.so
# A foo specific ctrl.
some_ctrl = some_value
# Another ctrl that doesn't take a value.
other_ctrl = EMPTY
# Supply all default algorithms
default_algorithms = ALL
```

6.2.6 Конфигурационный модуль для EVP

Этот модуль имеет имя alg_section, которое указывает на секцию, содержащую команды алгоритма. В настоящее время единственной поддерживаемой командой алгоритма является fips_mode, чье значение должно быть логической строкой, такой как on или off. Если значение on, то производится попытка входа в режим FIPS. Если запрос не проходит или библиотека не поддерживает режим FIPS, то возникает ошибка. Например:

```
alg_section = evp_settings

[evp_settings]

fips_mode = on
```

6.2.7 Конфигурационный модуль для SSL

Этот модуль имеет имя ssl_conf, которое указывает на секцию, содержащую конфигурации SSL.

Каждая строка в секции конфигурации SSL содержит имя конфигурации и имя секции, в которой она содержится. Каждая секция конфигурации состоит из пар команда-значение для SSL_CONF. Каждая пара будет передана в структуру SSL_CTX или SSL, если она вызывает

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

SSL_CTX_config() или SSL_config() с соответствующим именем конфигурации. Замечание: любые символы до первой точки в секции конфигурации игнорируются, поэтому одна команда может использоваться несколько раз. Например:

```
ssl_conf = ssl_sect

[ssl_sect]

server = server_section

[server_section]

RSA.Certificate = server-rsa.pem
ECDSA.Certificate = server-ecdsa.pem
Ciphers = ALL:!RC4
```

6.2.8 Примечания

Если конфигурационный файл пытается подставить несуществующую переменную, выставляется флаг ошибки и файл не будет загружен. Это может произойти, если делается попытка подставить несуществующую переменную окружения. Например, в предыдущей версии OpenSSL умолчательный главный конфигурационный файл OpenSSL использовал значение HOME, которое не могло быть определено в не-Unix-подобных операционных системах, что вызывало ошибку.

Это можно обойти, включив умолчательную секцию, которая предоставляет умолчательное значение: тогда если в окружении не удастся найти соответствующую строку, вместо нее будет использовано умолчательное значение. Чтобы это надежно работало, умолчательное значение нужно определить в конфигурационном файле до подстановки. В разделе 6.2.9 приведен пример, как это следует делать.

Если в одной и той же секции существует несколько значений одной и той же переменной, то все значения, кроме последнего, будут проигнорированы. В некоторых обстоятельствах, например с DN, одно и то же поле может встретиться несколько раз. Это обычно обходится игнонироваением любых символов перед первой . Например:

```
1.OU="My first OU"
2.OU="My Second OU"
```

Указывать в конфигурационном файле ключевые файлы возможно только в случае, если соответствующая команда утилиты openssl (из описанных в данном руководстве это только команда ca) поддерживает указание ключей в конфигурационном файле. При этом, если ключ расположен не в файле, а на аппаратном носителе, то указание на этот факт (-keyform engine) должно присутствовать в командной строке.

6.2.9 Примеры

Здесь приведен образец конфигурационного файла с использованием некоторых возможностей, описанных выше.

```
# This is the default section.
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
HOME=/temp
RANDFILE= ${ENV::HOME}/.rnd
configdir=$ENV::HOME/config

[ section_one ]

# We are now in section one.

# Quotes permit leading and trailing whitespace
any = " any variable name "

other = A string that can \
cover several lines \
by including \\ characters

message = Hello World\n

[ section_two ]

greeting = $section_one::message
```

Следующий пример показывает, как безопасно подставлять переменные окружения.

Предположим, вы хотите, чтобы переменная tmpfile указывала на временное имя файла. Каталог, в котором помещен файл, можно определить с помощью переменных окружения TEMP или TMP, но им может не быть присвоено вообще никакого значения. Если вы просто включите названия переменных окружения, а какая-нибудь из этих переменных не существует, это вызовет ошибку при попытке загрузить конфигурационный файл. При использовании умолчательной секции можно найти обе переменные, причем TEMP будет иметь приоритет, а если ни одна из них не определена, будет использована /textbackslash tmp:

```
TMP=/tmp
# The above value is used if TMP isn't in the environment
TEMP=$ENV::TMP
# The above value is used if TEMP isn't in the environment
tmpfile=${ENV::TEMP}/tmp.filename
```

Пример простой конфигурации библиотеки OpenSSL для входа в режим FIPS:

```
# Default appname: should match "appname" parameter (if any)
# supplied to CONF_modules_load_file et al.
openssl_conf = openssl_conf_section

[openssl_conf_section]
# Configuration module list
alg_section = evp_sect

[evp_sect]
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
# Set to "yes" to enter FIPS mode if supported
fips_mode = yes
```

Замечание: в приведенном выше примере вы получите ошибку с версиями OpenSSL, не поддерживающими FIPS. Более сложная конфигурация библиотеки OpenSSL. Добавить OID и не входить в режим FIPS:

```
# Default appname: should match "appname" parameter (if any)
# supplied to CONF_modules_load_file et al.
openssl_conf = openssl_conf_section

[openssl_conf_section]
# Configuration module list
alg_section = evp_sect
oid_section = new_oids

 [evp_sect]
# This will have no effect as FIPS mode is off by default.
# Set to "yes" to enter FIPS mode, if supported
fips_mode = no

[new_oids]
# New OID, just short name
newoid1 = 1.2.3.4.1
# New OID shortname and long name
newoid2 = New OID 2 long name, 1.2.3.4.2
```

Приведенный выше пример может использоваться с любым приложением, поддерживающим конфигурацию библиотеки, если "openssl_conf"изменено в соответствии с "appname".

Пример, когда приведенный выше второй файл примера сохраняется в "example.cnf тогда должна использоваться командная строка:

```
OPENSSL_CONF=example.cnf openssl asn1parse -genstr OID:1.2.3.4.1
```

будет выведено:

```
0:d=0  hl=2 l=   4 prim: OBJECT            :newoid1
```

показано, что OID "newoid1"был добавлен как "1.2.3.4.1".

6.2.10 Формат конфигураций расширений сертификатов

6.2.10.1 Описание

Несколько утилит библитеки OpenSSL могут добавлять расширения к сертификатам или запросам на сертификаты на основе содержания конфигурационного файла.

Как правило, приложение содержит опцию, указывающую на секцию расширений. Каждая строка секции расширений принимает форму:

```
extension_name=[critical,] extension_options
```

Если присутствует опция critical, расширение будет критическим.

Формат значений extension_options зависит от значения extension_name.

Есть четыре главных типов расширений: строковые расширения, многозначные расширения, бинарные и произвольные расширения.

Строковые расширения включают просто строку символов, содержащую или само значение или как его получить.

Например:

```
nsComment="This is a Comment"
```

Многозначные расширения имеют короткую и длинную форму. Короткая форма — это список имен и значений:

```
basicConstraints=critical,CA:true,pathlen:1
```

Длинная форма позволяет поместить значения в отдельную секцию:

```
basicConstraints=critical,@bs_section

[bs_section]

CA=true
pathlen=1
```

Обе формы эквивалентны.

Синтаксис бинарных расширений определяется кодом расширения: например, оно может содержать данные в нескольких секциях. Правильный синтаксис в каждом случае определяется самим кодом расширения: для примера просмотрите расширение, содержащее политику сертификата.

Если какой-то тип расширения не поддерживается, следует пользоваться синтаксисом произвольных расширений, см. ниже раздел 6.2.10.4.

6.2.10.2 Стандартные расширения

Следующие разделы в подробностях описывают каждое поддерживаемое расширение.

6.2.10.2.1 Basic Constraints

Это многозначное расширение, которое указывает, является ли сертификат сертификатом УЦ. Первое (обязательное) имя — CA, за ним следует TRUE или FALSE. Если значение CA равно TRUE, то можно включить опциональное имя pathlen, за которым следует неотрицательное число.

Примеры:

```
basicConstraints=CA:TRUE

basicConstraints=CA:FALSE

basicConstraints=critical,CA:TRUE, pathlen:0
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Сертификат УЦ должен включать значение basicConstraints, где значение поля CA установлено в TRUE. В конечном пользовательском сертификате следует или установить поле CA в значение FALSE, или полностью исключить это расширение. Некоторые приложения могут требовать включение basicConstraints с полем CA, установленным в значение FALSE, в пользовательские сертификаты.

Параметр pathlen указывает максимальное количество сертификатов УЦ, которое может появляться под этим сертификатом в цепочке. Поэтому если у вас сертификат УЦ со значением pathlen, равным нулю, то его можно использовать только для подписи конечных пользовательских сертификатов, но не других сертификатов УЦ.

6.2.10.2.2 Key Usage

Key Usage — это многозначное расширение, состоящее из списка имен дозволенных применений ключа.

Поддерживаемые имена: digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly and decipherOnly.

Примеры:

```
keyUsage=digitalSignature, nonRepudiation

keyUsage=critical, keyCertSign
```

6.2.10.2.3 Extended Key Usage

Это расширение состоит из списка применений, указывающих цели, для которых можно использовать открытый ключ, содержащийся в сертификате.

Они могут быть либо короткими именами объектов или численно-точечной формой OIDов. Хотя можно использовать любой OID, смысл имеют только некоторые значения. Особенно важны следующие значения PKIX, NS и MS:

```
Значение              Смысл
-----                 -------
 serverAuth            SSL/TLS Web Server Authentication.
 clientAuth            SSL/TLS Web Client Authentication.
 codeSigning           Code signing.
 emailProtection       E-mail Protection (S/MIME).
 timeStamping          Trusted Timestamping
 OCSPSigning           OCSP Signing
 ipsecIKE              ipsec Internet Key Exchange
 msCodeInd             Microsoft Individual Code Signing (authenticode)
 msCodeCom             Microsoft Commercial Code Signing (authenticode)
 msCTLSign             Microsoft Trust List Signing
 msEFS                 Microsoft Encrypted File System
```

Примеры:

```
extendedKeyUsage=critical,codeSigning,1.2.3.4
extendedKeyUsage=nsSGC,msSGC
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

6.2.10.2.4 Subject Key Identifier

Это на самом деле строкоеове расширение, которое может принимать два возможных значения. Или слово hash, которое автоматически соответствует указаниям из RFC3280, или шестнадцатиричную строку, задающую значение расширения, которое следует использовать. Использование шестнадцатиричное строки не рекомендуется.

Пример:

```
subjectKeyIdentifier=hash
```

6.2.10.2.5 Authority Key Identifier

Расширение Authority Key Identifier имеет два компонента: keyid и issuer; оба могут принимать опциональное значение «always».

Если присутствует компонент keyid, делается попытка скопировать Subject Key Identifier из родительского сертификата. Если присутствует компонент always, то при неудачном завершении выполнения этой попытки возвращается ошибка.

Компонент issuer копирует issuer и serial number из сертификата issuer. Это выполняется только в том случае, если компонент keyid выполнить не удалось, или компонент keyid не включен, но флаг always всегда будет включать это значение.

Пример:

```
authorityKeyIdentifier=keyid,issuer
```

6.2.10.2.6 Subject Alternative Name

Расширение subject alternative name позволяет включать различные буквенные значения в конфигурационный файл. Они включают email (адресс электронной почты), URI (единый индикатор ресурсов), DNS (DNS-имя домена), RID (зарегистрированный ID: OBJECT IDENTIFIER), IP (IP-адрес), dirName (distinguished name) и otherName.

Компонент email включает специальное значение copy. Это автоматически включает все электронные адреса, содержащиеся в поле сертификата subject name, в расширение.

IP-адрес, использующийся в компоненте IP, может быть или в формате IPv4, или в формате IPv6.

Значение dirName должно указывать на секцию, содержащую необходимое distinguished name в виде набора пар имя-значение. Прибавив спереди к имени знак +, можно сформировать многозначные AVA.

Значение otherName может включать произвольные данные, связанные с OID: значением должен быть OID, за которым следует точка с запятой и содержание в стандартном формате, воспринимаемом функцией ASN1_generate_nconf.

Примеры:

```
subjectAltName=email:copy,email:my@other.address,URI:http://my.url.here/
subjectAltName=IP:192.168.7.1
subjectAltName=IP:13::17
subjectAltName=email:my@other.address,RID:1.2.3.4
subjectAltName=otherName:1.2.3.4;UTF8:some other identifier

subjectAltName=dirName:dir_sect
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
[dir_sect]
C=UK
O=My Organization
OU=My Unit
CN=My Name
```

6.2.10.2.7 Issuer Alternative Name

Компонент issuer alternative name поддерживает все буквенные опции subject alternative name. Он **не** поддерживает опцию email:copy, потому что это не имело бы смысла. Но он поддерживает дополнительную опцию issuer:copy, которая копирует все значения subject alternative name из сертификата issuer (если возможно).

Пример:

```
issuserAltName = issuer:copy
```

6.2.10.2.8 Authority Info Access

Расширение authority information access предоставляет подробное описание того, как получить определенную информацию, связанную с сертификатом УЦ. Его синтаксис - accessOID;location, где location имеет тот же синтаксис, что и subject alternative name (только email:copy не поддерживается). accessOID может быть любым действительным OID, но только некоторые значения имеют смысл, например OCSP и caIssuers.

Пример:

```
authorityInfoAccess = OCSP;URI:http://ocsp.my.host/
authorityInfoAccess = caIssuers;URI:http://my.ca/ca.html
```

6.2.10.2.9 CRL distribution points

Это многозначное расширение, компоненты которого можно указать или в виде пар имя:значение, используя ту же форму, что и у subject alternative name, или в виде одного значения, представляющего собой имя секции, содержащей все поля точки распределения CRL.

Если очередной компонент DistributionPoint имеет формат пары имя:значение, то создается значение с полем fullName, установленным в данное значение, и отсутствующими полями cRLissuer и reasons.

В случае, если компонент имеет форму имени секции, то указанная секция содержит значения для каждого поля. В этой секции:

Если значение поля «имя» — fullname, то поле «значение» должно содержать полное имя точки распределения в том же формате, что и subject alternative name.

Если значение поля «имя» — relativename, то поле «значение» должно содержать имя секции, содержание которой представляет собой фрагмент Distinguished Name, который следует поместить в это поле.

Если присутствует имя CRLIssuer, оно должно содержать значение для этого поля в формате subject alternative name.

Если значение поля «имя» — reasons, то поле «значение» должно состоять из разделенных запятыми полей, содержащих причины. Поддерживаются причины: «keyCompromise»,

«CACompromise», «affiliationChanged», «superseded», «cessationOfOperation», «certificateHold», «privilegeWithdrawn» and «AACompromise».

Простые примеры:

```
crlDistributionPoints=URI:http://myhost.com/myca.crl
crlDistributionPoints=URI:http://my.com/my.crl,URI:http://oth.com/my.crl
```

Полный пример точки распределения:

```
crlDistributionPoints=crldp1_section

[crldp1_section]

fullname=URI:http://myhost.com/myca.crl
CRLissuer=dirName:issuer_sect
reasons=keyCompromise, CACompromise

[issuer_sect]
C=UK
O=Organisation
CN=Some Name
```

6.2.10.2.10 Issuing Distribution Points

Это расширение должно появляться только в списках отзыва сертификатов (CRL). Это многозначное расширение, синтаксис которого похож на синтаксис «секции», на которую указывает расширение CRL distribution points, но с некоторыми отличиями.

Имена reasons и CRLissuer не распознаются.

Допустимо имя onlysomereasons, которое устанавливает значение поля reasons. Значение указываетя в том же формате, что и поле reasons расширения CRL distribution point.

Также допускаются имена «onlyuser», «onlyCA», «onlyAA» and «indirectCRL». Значения должны быть булевыми (TRUE или FALSE), чтобы указанть значение соответствующего поля.

Пример:

```
issuingDistributionPoint=critical, @idp_section

[idp_section]

fullname=URI:http://myhost.com/myca.crl
indirectCRL=TRUE
onlysomereasons=keyCompromise, CACompromise

[issuer_sect]
C=UK
O=Organisation
CN=Some Name
```

6.2.10.2.11 Certificate Policies

Это бинарное расширение. Все поля этого расширения могут быть установлены с помощью соответствующего синтаксиса.

Если вы следуете рекомендациям PKIX и просто используете один OID, вам следует просто включить значение этого OID. Более одного OID можно установить, разделив их запятыми, например:

```
certificatePolicies= 1.2.4.5, 1.1.3.4
```

Если вы хотите включить квалификаторы, то OID политики и квалификаторы должны быть указаны в отдельной секции: это делается с использованием синтаксиса @section вместо буквенного значения OID.

Секция, на которую указывают, должна включать OID политики с использованием имени policyIdentifier, квалификаторы cPSuri могут быть включены с помощью синтаксиса:

```
CPS.nnn=value
userNotice.nnn=@notice
```

Значение квалификатора userNotice указывается в соответствующей секции. Эта секция может включать компоненты explicitText, organization и noticeNumbers. explicitText и organization — текстовые строки, noticeNumbers — список чисел, разделенных запятыми. Если включены компоненты organization и noticeNumbers, они должны присутствовать **оба**. Если вы используете компонент userNotice с Internet Explorer 5, вам понадобится опция ia5org на верхнем уровне, чтобы модифицировать кодировку; в противном случае она не будет правильно интерпретирована.

Пример:

```
certificatePolicies=ia5org,1.2.3.4,1.5.6.7.8,@polsect

[polsect]

policyIdentifier = 1.3.5.8
CPS.1="http://my.host.name/"
CPS.2="http://my.your.name/"
userNotice.1=@notice

[notice]

explicitText="Explicit Text Here"
organization="Organisation Name"
noticeNumbers=1,2,3,4
```

Опция ia5org изменяет тип поля organization. В RFC2459 это поле может быть только типа DisplayText. В RFC3280 также допустим тип IA5Strring. Некоторые приложения (например, некоторые версии Internet Explorer) могут потребовать ia5org.

6.2.10.2.12 Policy Constraints

Это многозначное расширение, состоящее из имен requireExplicitPolicy или inhibitPolicyMapping и неотрицательного целого числа. Должен присутствовать по крайней мере один компонент.

Пример:

```
policyConstraints = requireExplicitPolicy:3
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

6.2.10.2.13 Inhibit Any Policy

Это строковое выражение, значением которого должно быть неотрицательное целое число.
Пример:

```
inhibitAnyPolicy = 2
```

6.2.10.2.14 Name Constraints

Расширение name constraints — многозначное расширение. Имя должно начинаться со слов permitted или excluded, за которыми следует точка с запятой. Остальная часть имени и значение следует синтаксису расширения subjectAltName, за исключением того, что email:copy не поддерживается, и форма IP должна состоять из IP-адресов и маски подсети, разделенные символом /.
Примеры:

```
nameConstraints=permitted;IP:192.168.0.0/255.255.0.0

nameConstraints=permitted;email:.somedomain.com

nameConstraints=excluded;email:.com
```

6.2.10.2.15 OCSP No Check

Расширение OCSP No Check — это строковое расширение, но его значение устанавливается в ignored.
Пример:

```
noCheck = ignored
```

6.2.10.2.16 Опция TLS (также известная как Must Staple)

Это многозначное расширение, состоящее из списка идентификаторов расширений TLS. Каждый из идентификаторов может быть числом (от 0 до 65535) или поддерживаемым именем. Когда клиент TLS посылает расширение из списка, то ожидается, что сервер TLS включит это расширение в ответ. Поддерживаемые имена: status_request и status_request_v2.
Пример:

```
tlsfeature = status_request
```

6.2.10.3 Нерекомендуемые расширения

Описанные ниже расширения не являются стандартными, они специфичны для Netscape и в основном устарели. Их использование в новых приложениях не рекомендуется

6.2.10.3.1 Расширения Netscape String

Netscape Comment (nsComment) - строковое расширение, содержащее комментарий, который выводится в некоторых браузерах при просмотре сертификата.

Пример:

```
nsComment = "Some Random Comment"
```

Другие поддерживаемые расширения из этой категории: nsBaseUrl, nsRevocationUrl, nsCaRevocationUrl, nsRenewalUrl, nsCaPolicyUrl и nsSslServerName.

6.2.10.3.2 Netscape Certificate Type

Это многозначное расширение, состоящее из списка флагов, который нужно включить в сертификат. Оно использовалось для указания целей, для которых можно использовать сертификат. Сейчас вместо него используются расширения basicConstraints, keyUsage и extended key usage.

Допустимые значения для расширения nsCertType: client, server, email, objsign, reserved, sslCA, emailCA, objCA.

6.2.10.4 Произвольные расширения

Если расширение не поддерживается в коде OpenSSL, его следует закодировать, используя формат произвольных расширений. Можно также использовать формат произвольных расширений и для поддерживаемых расширений. Следует обращать особое внимание на то, чтобы данные были отформатированы корректно в соответствии с типом данного расширения.

Существует два способа кодирования произвольных расширений.

Первый способ - использовать слово ASN1, за которым следует содержание расширения в формате, воспринимаемом функцией ASN1_generate_nconf. Например:

```
1.2.3.4=critical,ASN1:UTF8String:Some random data

1.2.3.4=ASN1:SEQUENCE:seq_sect

[seq_sect]

field1 = UTF8:field1
field2 = UTF8:field2
```

Возможно также использовать слово DER для включения бинарных кодированных данных в любое расширение.

```
1.2.3.4=critical,DER:01:02:03:04
1.2.3.4=DER:01020304
```

Значение, следующее за DER — шестнадцатиричный дамп DER-кодировки расширения. Любое расширение можно записать в этой форме, чтобы переопределить поведение по умолчанию. Например:

```
basicConstraints=critical,DER:00:01:02:03
```

6.2.10.5 Предупреждение

Нет гарантии, что конкретная реализация будет обрабатывать каждое конкретное расширение. Таким образом, иногда существует возможность использовать сертификаты в целях, запрещенных в их расширениях, потому что конкретное приложение не распознает или не учитывает значения соответствующих расширений.

Необходимо с осторожностью использовать опции DER и ASN1. Если неосторожно использовать эти опции, можно создать совершенно некорректные расширения.

6.2.10.6 Примечания

Если расширение многозначно, и поле «значение» должно содержать запятую, следует использовать длинную форму, иначе запятая может быть неправильно интерпретирована как разделитель полей. Например:

```
subjectAltName=URI:ldap://somehost.com/CN=foo,OU=bar
```

вызовет ошибку, но эквивалентная форма:

```
subjectAltName=@subject_alt_section

[subject_alt_section]
subjectAltName=URI:ldap://somehost.com/CN=foo,OU=bar
```

вполне корректна.

Из-за поведения библиотеки OpenSSL conf одно и то же имя поля может встречаться только один раз в секции. Это означает, что:

```
subjectAltName=@alt_section

[alt_section]

email=steve@here
email=steve@there
```

распознает только последнее значение. Это можно обойти, используя форму:

```
[alt_section]

email.1=steve@here
email.2=steve@there
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7 Использование

7.1 Формат вызова утилиты

openssl команда [опции команды] [аргументы команды] — общий формат вызова утилиты

openssl [list-standard-commands | list-message-digest-commands | list-cipher-commands | list-cipher-algorithms | list-message-digest-algorithms | list-public-key-algorithms] — формат вызова псевдокоманд

openssl no-XXX [необязательные опции] — проверка существования команды

7.2 Использование алгоритмов ГОСТ

В большинстве случаев при использовании ключей «МагПро КриптоПакет» 3.0 в приложениях явное указание алгоритма не нужно — используемый алгоритм автоматически определяется на основе используемого ключа.

Необходимо явным образом указывать алгоритмы при использовании команд:

`dgst` (см. раздел 7.6.7) — для использования алгоритма хэширования по ГОСТ Р 34.11 всегда необходимо указывать параметр -md_gost12_256, -md_gost12_512 или -md_gost94 (последний вариант следует использовать только для обеспечения совместимости с предыдущими версиями);

`req` (см. раздел 7.6.18) — если использовать эту команду для создания ключей, необходимо явным образом указывать алгоритм gost2012_256, gost2012_512 или gost2001 с нужным набором параметров (алгоритм gost2001 следует использовать только для обеспечения совместимости с предыдущими версиями).

`smime` (см. раздел 7.6.22) и `cms` (см. раздел 7.6.4) — при использовании опции -encrypt следует указывать алгоритм симметричного шифрования -gost89;

Следует иметь в виду, что после 31 декабря 2018 года алгоритм ГОСТ Р 34.10-2001 должен использоваться только для проверки ранее выработанных подписей.

7.3 Ключи на аппаратных носителях

Пользователь утилиты openssl может воспользоваться ключами, находящимися на аппаратных носителях, если используемая команда имеет опцию keyform или аналогичную. В этом случае в качестве значения опции keyform указывается ENGINE, в качестве значения опции engine указывается cryptocom, а значение опции key строится следующим образом.

Вместо ключевого файла указывается строка-идентификатор, имеющая следующий формат: *ТИП-УСТРОЙСТВА[=ID][:имя-контейнера]*[.{X|S|P}

Здесь ID — специфичный для устройства аппаратный идентификатор, имя-контейнера - имя ключевого контейнера, заданное при его создании. Расширения .X и .S указываются только в случае хранения ключей на Вьюге: X — ключ обмена ключами, S — ключ подписи. Расширение .P используется только при создании ключевого контейнера и предписывает защиту контейнера PIN-кодом (только для носителей, поддерживающих такую возможность).

Попытка обращения к ключу с именем контейнера, отличным от того, который имеется на доступном в данный момент устройстве, приводит к ошибочному завершению операции.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.4 Пароли как аргументы

Некоторые команды принимают пароли в качестве аргументов, как правило, используя для входного и выходного паролей соответственно опции -passin и -passout. Эти опции позволяют получать пароли из различных источников. Каждая из этих опций принимает один аргумент, формат которого показан ниже. Если аргумент не указан, а пароль запрашивается, пользователю предлагается ввести пароль: как правило, такой пароль вводится с текущего терминала без вывода на экран.

Формат аргумента	**Описание**
pass:пароль	Пароль указывается явным образом. Поскольку такой пароль видят утилиты (например утилита ps под Unix-подобные ОС), этот способ ввода пароля следует использовать только в том случае, если безопасность не важна.
env:var	Считать пароль из переменной среды var. Поскольку среда других процессов на некоторых платформах видна (например, ps под некоторыми Unix-подобными ОС), этот способ ввода пароля следует использовать с осторожностью.
file:pathname	Первая строка файла с именем pathname является паролем. Если одно и то же имя файла указывается в качестве аргумента опций -passin и -passout, то для входного пароля будет использована первая строка файла, а для выходного — вторая. Необязательно указывать именно на файл: можно, например, указывать на устройство или именованный канал.
fd:number	Прочитать пароль из открытого файла текущего процесса, заданного числовым дескриптором number. Это можно использовать, например, чтобы передать данные по неименованному каналу.
stdin	Прочитать пароль со стандартного ввода.

Если пароль содержит русские буквы, он доолжен быть передан в кодировке UTF-8. Как следствие, корректное указание такого пароля в командрой строке или ввод его с клавиатуры возможны только если программа запущена в локали UTF-8.

7.5 Описание команд

Утилита openssl предоставляет широкий выбор команд (см. выше употребление понятия «команда» в формате вызова утилиты), многие из которых используются с различными опциями и аргументами (см. выше «опции команды» и «аргументы команды»).

Псевдокоманды list-standard-commands, list-message-digest-commands и list-cipher-commands выводят список имен (построчно) стандартных команд, список доступных команд хэширования и список доступных команд шифров.

Псевдокоманды list-cipher-algorithms и list-message-digest-algorithms выводят список доступных алгоритмов шифров и алгоритмов хэширования.

Псевдокоманда list-public-key-algorithms выводит список поддерживаемых алгоритмов с открытыми ключами.

Псевдокоманда no-XXX проверяет, доступна ли указанная команда (вместо XXX указывается название команды). Если команды с указанным именем не существует, команда no-XXX возвращает 0 (успех) и выводит no-XXX; в противном случае она возвращает 1 и выводит

XXX. В обоих случаях результат направляется в стандартный вывод и ничего не выводится в stderr. Дополнительные командно-строчные аргументы всегда игнорируются.

Команда no-XXX не может определить доступность псевдокоманд, таких как quit, а также самой команды no-XXX.

7.6 Стандартные команды

Команда	**Описание**
`asn1parse`	Разбор ASN.1-структур, понимаемых библиотекой
`ca`	Управление удостоверяющим центром
`ciphers`	Вывод списка доступных шифр-сьютов SSL/TLS
`cms`	Утилита CMS (Cryptographic Message Syntax)
`crl`	Обработка списком отзыва сертификатов (CRL)
`crl2pkcs7`	Создание специального сообщения формата `PKCS#7` из CRL и списка сертификатов
`dgst`	Вычисление хэш-суммы
`errstr`	Расшифровка кода ошибки
`genpkey`	Генерация закрытого ключа.
`genrsa`	Генерация закрытого ключа RSA
`ocsp`	Утилита онлайн-протокола статусов сертификатов.
`pkcs7`	Работа с форматом `PKCS#7`
`pkcs8`	Работа с форматом `PKCS#8`
`pkcs12`	Работа с форматом `PKCS#12`
`pkey`	Управление открытым и закрытым ключом.
`pkeyparam`	Управление параметром алгоритма открытого ключа.
`pkeyutl`	Утилита для выполнения криптографических операций с алгоритмами открытых ключей.
`rand`	Генерация псевдослучайных байтов
`req`	Работа с заявкой на получение сертификата формата X.509 (CSR)
`s_client`	Реализация SSL/TLS-клиента общего назначения, который может установить прозрачное соединение с удаленным сервером, работающим по протоколу SSL/TLS. Клиент предназначен только для тестовых целей и предоставляет только простейший интерфейс с рудиментарной функциональностью, хотя внутри себя он использует практически всю функциональность библиотеки OpenSSL.
`s_server`	Реализация SSL/TLS-сервера общего назначения, с которым могут быть установлены соединения удаленными клиентами, работающими по протоколу SSL/TLS. Сервер предназначен только для тестовых целей и предоставляет только простейший интерфейс с рудиментарной функциональностью, хотя внутри себя он использует практически всю функциональность библиотеки OpenSSL. Он предоставляет как свой собственный, ориентированный на работу в командной строке протокол для тестирования функций SSL, так и простейший http-сервер.
`s_time`	Измеритель производительности SSL.
`smime`	Обработка почтовых сообщений формата `S/MIME`
`speed`	Определение скорости работы алгоритма

Команда	Описание
`spkac`	Утилита генерации и вывода SPKAC (Signed Public Key and Challenge)
`ts`	Инструмент для работы с метками времени TSA (клиент/сервер).
`verify`	Проверка корректности сертификатов формата X.509
`version`	Вывод информации о версии библиотеки OpenSSL
`x509`	Работа с сертификатами формата X.509

7.6.1 Команда `asn1parse`

7.6.1.1 Описание команды

Команда asn1parse — диагностическая утилита, которая может интерпретировать ASN.1-структуры. Ее также можно использовать для чтения данных, записанных в формате ASN.1.

7.6.1.2 Формат ввода команды

openssl asn1parse [-inform PEM|DER] [-in filename] [-out filename] [-noout] [-offset number] [-length number] [-i] [-oid filename] [-dump] [-dlimit num] [-strparse offset] [-genstr string] [-genconf file]

7.6.1.3 Опции команды

Опция	Описание
-inform DER|PEM	Формат входных данных. DER — двоичный формат данных, а PEM (умолчание) — данные в кодировке base64.
-in filename	входной файл, стандартный ввод по умолчанию
-out filename	выходной файл, в который записываются данные в DER-кодировке. Если эта опция отсутствует, не выводится никаких данных. Это особенно полезно в сочетании с опцией -strparse.
-noout	Отменяет вывод интерпретированной версии входного файла
-offset number	Начальное смещение при интерпретации, по умолчанию — начало файла
-length number	Количество байтов, которые следует интерпретировать, по умолчанию — до конца файла.
-i	Выводит данные с отступами, соответствующими «глубине» структур
-oid filename	Файл, содержащий дополнительные OBJECT IDENTIFIER-ы (OID-ы). Формат этого файла описан ниже в разделе 7.6.1.5.

Опция	Описание
-dump	Отображает данные неизвестного типа в шестнадцатеричном формате
-dlimit num	Аналогично –dump, но выдаются только первые указанные num байт.
-strparse offset	Интерпретирует октеты, содержащиеся в ASN.1-объекте, начиная с позиции offset. Эту опцию можно использовать несколько раз, чтобы проанализировать вложенную структуру.
-genstr string, -genconf file	Генерирует зашифрованные данные, основанные на значениях string, file или и том и другом с использованием стандартного формата, воспринимаемого функцией ASN1_generate_nconf. Если присутствует только file, то значение string берется из умолчательной секции с использованием имени asn1. Зашифрованные данные пропускаются через интерпретатор ASN1 и выводятся, как выводились бы данные из файла, таким образом, содержание может быть просмотрено и записано в файл с помощью опции out.

7.6.1.4 Вывод

Вывод, как правило, содержит строки, подобные следующим:

```
0:d=0  hl=4 l= 681 cons: SEQUENCE
       .....
229:d=3  hl=3 l= 141 prim: BIT STRING
373:d=2  hl=3 l= 162 cons: cont [ 3 ]
376:d=3  hl=3 l= 159 cons: SEQUENCE
379:d=4  hl=2 l=  29 cons: SEQUENCE
381:d=5  hl=2 l=   3 prim: OBJECT        :X509v3 Subject Key Identifier
386:d=5  hl=2 l=  22 prim: OCTET STRING
410:d=4  hl=2 l= 112 cons: SEQUENCE
412:d=5  hl=2 l=   3 prim: OBJECT        :X509v3 Authority Key Identifier
417:d=5  hl=2 l= 105 prim: OCTET STRING
524:d=4  hl=2 l=  12 cons: SEQUENCE
       .....
```

Этот пример - часть самоподписанного сертификата. Каждая строка начинается с начального смещения в десятичной системе счисления. d=XX указывает текущую глубину. Глубина увеличивается в пределах любого SET или SEQUENCE. hl=XX указывает длину заголовка (октеты tag и length) текущего типа. l=XX указывает длину октетов содержания.

Можно использовать опцию -i, чтобы вывод читался легче.

Необходимо некоторое знание ASN.1-структуры, чтобы интерпретировать вывод.

В этом примере BIT STRING при начальном смещении 229 — это открытый ключ, содержащийся в сертификате. Октеты его содержания будут содержать информацию об открытом ключе. Если воспользоваться опцией -strparse 229, получим:

```
0:d=0  hl=3 l= 137 cons: SEQUENCE
3:d=1  hl=3 l= 129 prim: INTEGER        :E5D21E1F5C8D208EA7A2166C
7FAF9F6BDF2059669C60876DDB70840F1A5AAFA59699FE471F379F1DD6A487E7D
5409AB6A88D4A9746E24B91D8CF55DB3521015460C8EDE44EE8A4189F7A7BE77D
6CD3A9AF2696F486855CF58BF0EDF2B4068058C7A947F52548DDF7E15E96B385F
86422BEA9064A3EE9E1158A56E4A6F47E5897
135:d=1  hl=2 l=   3 prim: INTEGER           :010001
```

7.6.1.5 Примечания

Если OID не является частью внутренней таблицы OpenSSL, он будет представлен в численном виде (например 1.2.3.4). Файл, переданный в опцию -oid, позволяет включить дополнительные OID-ы. Каждая строка состоит из трех колонок, в первой колонке содержится OID в численной форме, за ним должен следовать пробел. Вторая колонка — «короткое имя», которое представляет собой одно слово, за которым следует пробел. Последняя колонка — остаток строки, представляющий собой «длинное имя». Команда asn1parse выводит длинное имя. Например:

```
1.2.3.4  shortName A long name
```

7.6.1.6 Примеры

Интерпретировать файл:

```
openssl asn1parse -in file.pem
```

Интерпретировать файл в DER-кодировке:

```
openssl asn1parse -inform DER -in file.der
```

Генерировать простую строку в кодировке UTF-8:

```
openssl asn1parse -genstr 'UTF8:Hello World'
```

Генерировать и вывести строку в кодировке UTF-8, не выводить интерпретированный вывод:

```
openssl asn1parse -genstr 'UTF8:Hello World' -noout -out utf8.der
```

Генерировать файл с использованием конфигурационного файла:

```
openssl asn1parse -genconf asn1.cnf -noout -out asn1.der
```

Пример конфигурационного файла:

```
asn1=SEQUENCE:seq_sect

[seq_sect]

field1=BOOL:TRUE
field2=EXP:0, UTF8:some random string
```

7.6.2 Команда ca

7.6.2.1 Описание команды

Команда ca реализует простейший удостоверяющий центр. Ее можно использовать для подписывания заявок на сертификаты различным образом и для генерации списков отзыва

сертификатов (CRL). Команда также поддерживает текстовую базу данных выпущенных сертификатов и их статуса.

7.6.2.2 Формат ввода команды

openssl ca [-verbose] [-config filename] [-name section] [-gencrl] [-revoke file] [-status serial] [-updatedb] [-crl_reason reason] [-crl_hold instruction] [-crl_compromise time] [-crl_CA_compromise time] [-crldays days] [-crlhours hours] [-crlexts section] [-startdate date] [-enddate date] [-days arg] [-md arg] [-policy arg] [-keyfile arg] [-keyform PEM|DER] [-key arg] [-passin arg] [-cert file] [-selfsign] [-in file] [-out file] [-notext] [-outdir dir] [-infiles] [-spkac file] [-ss_cert file] [-preserveDN] [-noemailDN] [-batch] [-msie_hack] [-extensions section] [-extfile section] [-engine id] [-subj arg] [-utf8] [-multivalue-rdn]

7.6.2.3 Опции команды

Описание опций разбито на разделы по их целевому назначению.

7.6.2.3.1 Опции обработки заявок и выпуска сертификатов

Опция	**Описание**
-config filename	Указывает, какой конфигурационный файл следует использовать.
-name section	Указывает, какой раздел конфигурационного файла использовать (имеет больший приоритет, нежели значение параметра default_ca в разделе ca файла конфигурации).
-in filename	Указывает входной файл, содержащий одну заявку на выдачу сертификата, которую следует подписать подписью удостоверяющего центра.
-ss_cert filename	Один самозаверенный сертификат, который следует подписать подписью удостоверяющего центра.
-spkac filename	Файл, содержащий один подписанный открытый ключ и challenge в формате Netscape и значения дополнительных полей, на основе которого следует выработать сертификат. См. раздел 7.6.24 для получения информации по требуемому формату.
-infiles	Если эта опция присутствует, она должна быть указана последней, все последующие аргументы считаются именами файлов, содержащих заявки на сертификаты.
-out filename	Выходной файл для сертификатов. По умолчанию — стандартный вывод. Информация о сертификате также будет выведена в этот файл.
-outdir directory	Выходной каталог для сертификатов. Сертификат будет записан в файл с именем, состоящим из серийного номера в шестнадцатеричном виде, с расширением `.pem`.
-cert file	Имя файла сертификата удостоверяющего центра.
-keyfile filename	Имя файла, содержащего закрытый ключ, с помощью которого следует вырабатывать подписи под сертификатами.

Опция	Описание
-keyform PEM|DER	Определяет формат данных в файле закрытого ключа. По умолчанию – PEM.
-key password	Пароль для зашифрования закрытого ключа. Поскольку в некоторых операционных системах аргументы командной строки видны (например Unix-подобные ОС с утилитой ps), эту опцию следует использовать с осторожностью.
-selfsign	Опция для перевыпуска сертификата удостоверяющего центра. Указывает, что выпускаемый сертификат следует подписать тем ключом, на котором были подписаны заявки (ключ определяется опцией -keyfile). Заявки, подписанные другим ключом, игнорируются. Если указаны опции -spkac, -ss_cert или -gencrl, опция -selfsign игнорируется. В результате использования опции -selfsign в базе данных сертификатов появляется самоподписанный сертификат, использующий тот же счетчик серийных номеров, что и другие сертификаты, подписанные с помощью самоподписанного сертификата.
-passin arg	Указывает, где содержится пароль для ключа. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4.
-verbose	Выводит дополнительную информацию о выполняемых операциях
-notext	Отменяет вывод текстовой формы сертификата в выходной файл.
-startdate date	Позволяет явным образом указать дату вступления сертификата в действие. Формат даты ГГММДДЧЧММССЗ (как в структуре ASN.1 UTCTime; З — временная зона).
-enddate date	Позволяет явным образом указать дату окончания действия сертификата. Формат даты ГГММДДЧЧММССЗ (как в структуре ASN.1 UTCTime; З — временная зона).
-days arg	Срок действия сертификата в днях.
-policy arg	Эта опция определяет, какая «политика» удостоверяющего центра используется. Это раздел конфигурационного файла, который определяет, какие поля должны быть обязательными или соответствовать сертификату удостоверяющего центра. Для получения дополнительной информации см. раздел 7.6.2.4.
-msie_hack	Это опция совместимости для поддержки работы удостоверяющего центра с очень старыми версиями Microsoft Internet Explorer. Так как использование российских криптоалгоритмов с этими старыми версиями IE невозможно, этой опцией пользоваться не следует.
-preserveDN	Обычно порядок полей в Distinguished Name сертификата совпадает с порядком полей в соответствующем разделе политики. Когда установлена эта опция, порядок полей такой же, как в заявке. Это делается в основном для совместимости со старыми версиями IE.

Опция	Описание
-noemailDN	Distinguished Name сертификата может содержать поле EMAIL, если такое поле присутствует в заявке, но считается хорошей политикой просто указывать адрес электронной почты в расширении сертификата altName. Когда установлена эта опция, поле EMAIL удаляется из тела сертификата и устанавливается только в присутствующих расширениях. Для достижения этого же результата можно также использовать ключевое слово email_in_dn в конфигурационном файле.
-batch	Эта опция устанавливает пакетный (неинтерактивный) режим работы. В этом режиме не задается никаких вопросов.
-extensions section	Указывает, что при выпуске сертификата необходимо добавить расширения, указанные в соответствующем разделе конфигурационного файла (опция по умолчанию для X509_extensions, если не используется опция -extfile). Если в момент выпуска сертификата в конфигурационном файле отсутствует раздел расширений, создается сертификат версии V1. Если раздел расширений присутствует (даже если он пуст), создается сертификат версии V3.
-extfile file	Прочитать расширения сертификата из дополнительного конфигурационного файла (с использованием умолчательного раздела, если не указана также опция -extensions).
-engine id	Указывает на загружаемый модуль engine (по его уникальной идентификационной строке) и вызывает попытку использовать реализацию криптографических алгоритмов из этого модуля.
-subj arg	Замещает содержание поля subject name, указанное в заявке. Формат аргумента arg должен быть /type0=value0/type1=value1/type2=..., символы могут быть экранированы знаком \(обратный слэш), пробелы не опускаются.
-utf8	Эта опция указывает, что значения полей следует интерпретировать как строки в кодировке UTF8, по умолчанию они интерпретируются в кодировке ASCII. Это означает, что значения полей, считанные с терминала или из конфигурационного файла, должны быть корректными UTF-8 строками.
-multivalue-rdn	Эта опция указывает, что аргумент опции -subj необходимо интерпретировать с полной поддержкой многозначных RDN. Пример: /DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe Если опция -multi-rdn не используется, то значение поля UID будет 123456+CN=John Doe.

7.6.2.3.2 Опции работы со списками отзыва сертификатов (CRL)

Опция	Описание
-gencrl	Эта опция генерирует список отзыва сертификатов на основе информации из индексного файла.

Опция	Описание
-crldays num	Срок действия списка отзыва сертификата в днях. Этот срок от момента выпуска будет указан в поле nextUpdate списка отзыва сертификатов.
-crlhours num	Срок действия списка отзыва сертификатов в часах.
-revoke filename	Файл, содержащий сертификат, который следует отозвать.
-status serial	Выводит статус сертификата с указанным серийным номером. Статус может быть valid, revoked, expired, suspended.
-updatedb	Обновляет базу данных, устанавливая статус expired для сертификатов, срок действия которых истек.
-crl_reason reason	Причина отзыва, где в качестве значения аргумента reason может быть: **unspecified** - не указана; **keyCompromise** - компрометация ключа владельца сертификата; **CACompromise** - компрометация ключа удостоверяющего центра; **affiliationChanged** - смена должности владельца сертификата; **superseded** - замена сертификата; **cessationOfOperation** - прекращение деятельности; **certificateHold** - временный отзыв сертификата; **removeFromCRL** - отмена отзыва ранее отозванного сертификата. Ключевые слова, указывающие причины отзыва, нечувствительны к регистру. Указание любой причины отзыва придаст сертификату версию V2. На практике причина removeFromCRL (отмена отзыва) не слишком широко употребляется, потому что она используется только в списках типа delta, которые сейчас не поддерживаются.
-crl_hold instruction	Эта опция устанавливает причину отзыва в certificateHold (временный отзыв сертификата) и значение аргумента instruction в инструкцию к временному отзыву, которая должна быть OID. Хотя можно использовать любой OID, обычно используются только holdInstructionNone (использование которого не рекомендовано в RFC2459), holdInstructionCallIssuer или holdInstructionReject.
-crl_compromise time	Эта опция устанавливает причину отзыва в keyCompromise (компрометация ключа владельца сертификата), а время компрометации — в значение аргумента time. Значение аргумента time должно быть указано в формате GeneralizedTime, т.е. ГГГГММДДЧЧММССЗ (где З — временная зона).
-crl_CA_compromise time	То же самое, что и -crl_compromise, с той разницей, что причина отзыва устанавливается в значение CACompromise.
-crlexts section	Раздел конфигурационного файла, содержащий расширения списков отзыва сертификатов, которые следует включить. Если в конфигурационном файле нет такого раздела, создается список версии V1, если присутствует (даже пустой), создается список версии V2. Указанные расширения являются расширениями именно CRL, а не его отдельных входов (описаний сертификатов). Следует отметить, что некоторые программы (например Netscape) не умеют работать со списками отзыва сертификатов версии V2.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.2.3.3 Опции конфигурационного файла

Раздел конфигурационного файла, содержащий опции для команды ca, находится следующим образом: если в командной строке указана опция -name, она указывает нужный раздел. В противном случае необходимый раздел должен быть указан в опции default_ca раздела ca конфигурационного файла (или умолчательного раздела конфигурационного файла). Кроме default_ca, непосредственно из раздела ca считываются следующие опции: RANDFILE, preserve, msie_hack. Кроме RANDFILE, это, вероятно, ошибка и может измениться в последующих версиях.

Многие опции конфигурационного файла идентичны опциям командной строки. Опции командной строки имеют приоритет над опциями конфигурационного файла. Если опция указана как обязательная, необходимо указать либо эту опцию в конфигурационном файле, либо ее аналог в командной строке (если такой аналог существует).

Опция	**Описание**
oid_file	Эта опция указывает на файл, содержащий дополнительные OID (OBJECT IDENTIFIERS). Каждая строка файла должна иметь следующий формат: OID в численном виде, пробел, короткое имя, пробел, длинное имя.
oid_section	Эта опция указывает на раздел конфигурационного файла, содержащий дополнительные OID. Каждая строка раздела должна иметь формат: короткое имя OID=численный вид OID. В случае использования этой опции короткое и длинное имена совпадают.
new_certs_dir	То же, что и опция командной строки -outdir. Указывает каталог, в который должны быть помещены новые сертификаты. Обязательна.
certificate	То же, что и опция командной строки -cert. Указывает файл, содержащий сертификат удостоверяющего центра. Обязательна.
private_key	То же, что и опция командной строки -keyfile. Указывает файл, содержащий закрытый ключ удостоверяющего центра. Обязательна.
RANDFILE	Файл, используемый для считывания и записи информации для инициализации генератора случайных чисел.
default_days	То же, что и опция командной строки -days. Срок действия сертификата в днях.
default_startdate	То же, что и опция командной строки -startdate. Дата начала действия сертификата. Если не установлена, используется текущее время.
default_enddate	То же, что и опция командной строки -enddate. Должна быть указана либо эта опция, либо опция default_days (или их командно-строчные эквиваленты).
default_crl_hours default_crl_days	То же, что и опции -crlhours и -crldays. Используются только в том случае, если не указано ни одной из соответствующих командно-строчных опций. Чтобы создать список отзыва сертификатов, необходимо указать хотя бы одну из этих четырех опций.
default_md	То же самое, что и опция -md. Алгоритм хэширования, который будет использован. Обязательная.

Опция	Описание
database	Используемый файл текстовой базы данных. Обязательная. Указанный файл должен существовать, хотя сначала он будет пустым.
unique_subject	Если указано значение этой опции yes, то описания сертификатов в базе данных должны иметь уникальные поля subject. Если указано значение no, несколько описаний актуальных сертификатов могут иметь одно и то же значение поля subject. Значение по умолчанию — yes для совместимости со старыми (до 0.9.8) версиями криптобиблиотеки OpenSSL. Однако для упрощения процедуры перевыпуска сертификата удостоверяющего центра рекомендуется использовать значение no, особенно в комбинации с опцией командной строки -selfsign.
serial	Текстовый файл, содержащий серийный номер для следующего сертификата в шестнадцатеричной форме. Обязательна. Указанный файл должен существовать и содержать корректный серийный номер.
crlnumber	Текстовый файл, содержащий номер для следующего списка отзыва сертификатов в шестнадцатеричной форме. Номер будет включен в списки отзывов сертификатов только в том случае, если этот файл существует. Если этот файл существует, он должен содержать корректный номер списка отзыва сертификатов.
x509_extensions	То же, что и опция командной строки -extensions.
crl_extensions	То же, что и опция командной строки -crlexts.
preserve	То же, что и опция командной строки -preserveDN
email_in_dn	То же, что и опция командной строки -noemailDN. Если вы хотите удалить поле EMAIL из distinguished name сертификата, просто установите значение этой опции в no. Если опция не указана, по умолчанию поле EMAIL в distinguished name сертификата позволено.
msie_hack	То же, что и опция командной строки -msie_hack
policy	То же, что и опция командной строки -policy. Обязательна. См. раздел 7.6.2.4 для получения дополнительной информации.
name_opt, cert_opt	Эти опции определяют формат вывода информации о сертификате при запросе подтверждения подписывания сертификата от пользователя. Здесь могут использоваться все опции, поддерживаемые опциями -nameopt и -certopt утилиты x509 (см. раздел 7.6.28), кроме того, что опции no_signame и no_sigdump жестко установлены и не могут быть отключены (потому что подпись под сертификатом не может быть выведена, т.к. в этот момент сертификат еще не подписан). Для удобства можно придать обеим этим опциям значения ca_default для получения достойного результата. Если обе эти опции отсутствуют, используется формат из старых версий криптобиблиотеки OpenSSL. Использование старого формата не рекомендуется, потому что он выводит только те поля, которые указаны в разделе политики, неправильно работает со строковыми типами данных и не выводит расширения.

Опция	Описание
copy_extensions	Эта опция определяет, как следует работать с расширениями в заявках на выдачу сертификата. Если значение опции установлено в none, или эта опция вообще не указана, расширения игнорируются и не переносятся в сертификат. Если значение этой опции установлено в copy, все расширения, имеющиеся в заявке, которых еще нет в сертификате, переносятся в сертификат. Если значение этой опции установлено в copyall, то все расширения из заявки переносятся в сертификат: если расширение уже присутствует в сертификате, оно сначала оттуда удаляется. Перед применением этой опции см. раздел 7.6.2.10. Главное применение этой опции — предоставить возможность переноса из заявки значений некоторых расширений, таких как subjectAltName.

7.6.2.4 Формат раздела политики

Раздел политики состоит из набора переменных, соответствующих полям distinguished name сертификата. Если значение переменной установлено в «match», то значение поля должно соответствовать тому же полю в сертификате удостоверяющего центра. Если значение установлено в «supplied», поле должно присутствовать. Если значение установлено в «optional», поле может присутствовать. Все поля, не упомянутые в разделе политики, удаляются без предупреждения, если только не указана опция -preserveDN, но этот случай можно рассматривать как отклонение от стандартного образа действий.

7.6.2.5 Формат SPKAC

Входными данными для опции командной строки -spkac являются подписанный открытый ключ и challenge формата Netscape. Обычно эти данные создаются тэгом KEYGEN html-формы, создающей новый закрытый ключ. Однако возможно создать SPKAC с помощью утилиты spkac.

Файл, указываемый в качестве аргумента опции -spkac, должен содержать переменную SPKAC, значение которой установлено в значение SPKAC, а также требуемые компоненты DN в виде пар "имя-значение". Если вам нужно включить один и тот же компонент дважды, перед ним можно указать номер с точкой.

Когда обрабатывается формат SPKAC, то вывод будет производиться либо в формате DER, если установлен флаг -out, либо в формате PEM, если вывод производится на stdout или установлен флаг –outdir.

7.6.2.6 Примеры

Примечание: эти примеры предполагают, что структура каталогов удостоверяющего центра уже создана и соответствующие файлы уже существуют. Это обычно включает создание сертификата удостоверяющего центра и закрытого ключа с помощью утилиты req, файла номера серийного файла и пустого индексного файла, и размещение их в соответствующих каталогах.

Чтобы использовать нижеуказанный пример конфигурационного файла, необходимо создать каталоги demoCA, demoCA/private и demoCA/newcerts. Сертификат удостоверяющего центра следует скопировать в файл demoCA/cacert.pem, а его закрытый ключ — в файл demoCA/private/cakey.pem. Следует создать файл demoCA/serial, содержащий, например, «01», и пустой индексный файл demoCA/index.txt.

Подписание заявки на сертификат:

```
openssl ca -in req.pem -out newcert.pem
```

Подписание заявки на сертификат с использованием расширений удостоверяющего центра:

```
openssl ca -in req.pem -extensions v3_ca -out newcert.pem
```

Создание списка отзыва сертификатов:

```
openssl ca -gencrl -out crl.pem
```

Подписывание нескольких заявок:

```
openssl ca -infiles req1.pem req2.pem req3.pem
```

Сертификация SPKAC формата Netscape:

```
openssl ca -spkac spkac.txt
```

Образец файла SPKAC (строка SPKAC сокращена для наглядности):

```
SPKAC=MIG0MGAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAn7PDhCeV/xIxUg8V70YRxK2A5
CN=Steve Test
emailAddress=steve@openssl.org
0.OU=OpenSSL Group
1.OU=Another Group
```

Образец конфигурационного файла с соответствующими разделами для команды ca:

```
[ ca ]
default_ca      = CA_default            # The default ca section

[ CA_default ]

dir             = ./demoCA              # top dir
database        = $dir/index.txt        # index file.
new_certs_dir   = $dir/newcerts         # new certs dir

certificate     = $dir/cacert.pem       # The CA cert
serial          = $dir/serial           # serial no file
private_key     = $dir/private/cakey.pem# CA private key
RANDFILE        = $dir/private/.rand    # random number file

default_days    = 365                   # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = md5                   # md to use

policy          = policy_any            # default policy
email_in_dn     = no                    # Don't add the email into cert DN

name_opt        = ca_default            # Subject name display option
cert_opt        = ca_default            # Certificate display option
copy_extensions = none                  # Don't copy extensions from request
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

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

7.6.2.7 Файлы

Примечание: Расположение всех файлов может быть изменено либо в опциях времени компилирования, либо в конфигурационном файле, переменных среди или опциях командной строки. Указанные значения являются умолчательными.

```
/usr/local/ssl/lib/openssl.cnf - главный конфигурационный файл
./demoCA                       - головной каталог удостоверяющего центра (УЦ)
./demoCA/cacert.pem            - сертификат удостоверяющего центра
./demoCA/private/cakey.pem     - Закрытый ключ удостоверяющего центра
./demoCA/serial                - Файл серийного номера УЦ
./demoCA/serial.old            - Бэкапный файл серийного номера УЦ
./demoCA/index.txt             - Файл текстовой базы данных УЦ
./demoCA/index.txt.old         - Бэкапный файл текстовой базы данных УЦ
./demoCA/certs                 - файл для вывода сертификатов
./demoCA/.rnd                  - информация для инициализации генератора
                                 случайных чисел УЦ
```

7.6.2.8 Переменные среды

Переменная среды `OPENSSL_CONF` отражает расположение главного конфигурационного файла. Опция командной строки -config имеет приоритет над этой переменной.

7.6.2.9 Ограничения

Индексный файл текстовой базы данных — критическая часть процесса, и если этот файл поврежден, его может оказаться трудно восстановить. Теоретически возможно восстановить индексный файл из всех выпущенных сертификатов и текущего списка отзыва сертификатов, но такой опции не существует.

Свойства версии V2 списков отзыва сертификатов, такие как дельта-списки, сейчас не поддерживаются.

Хотя одновременно может быть введено и обработано несколько заявок, можно включить только один SPKAC или самоподписанный сертификат.

7.6.2.10 Предупреждения

Команда ca прихотлива и иногда ведет себя недружественно по отношению к пользователю.

Утилита ca сначала была написана в качестве примера того, как работать с удостоверяющим центром. Она не планировалась как полнофункциональный удостоверяющий центр: однако некоторые используют ее в этом качестве.

Команда ca по сути представляет собой однопользовательскую команду: ни на какие файлы не накладываются блокировки, и попытки запустить более одной команды ca на одной и той же базе данных могут привести к непредсказуемым результатам.

Опцию copy_extensions следует использовать с осторожностью. Иначе может быть нарушена безопасность системы. Например, если заявка на сертификат содержит расширение basicConstraints с CA:TRUE, значение copy_extensions установлено в copyall, а пользователь этого не замечает, когда сертификат демонстрируется, это предоставит запрашивающему действительный сертификат удостоверяющего центра.

Этой ситуации можно избежать, установив опцию copy_extensions в copy и включив basicConstraints с CA:FALSE в конфигурационный файл. Тогда, если заявка будет содержать расширение basicConstraints, оно будет проигнорировано.

Также рекомендуется включать в конфигурационный файл значения других расширений, таких как keyUsage, чтобы предотвратить автоматический перенос значений этих расширений в сертификат из заявки.

Дополнительные ограничения можно включить в сам сертификат удостоверяющего центра. Например, если в сертификате удостоверяющего центра указано:

```
basicConstraints = CA:TRUE, pathlen:0
```

то даже если будет выпущен пользовательский сертификат с CA:TRUE, он будет недействителен.

7.6.3 Команда `ciphers`

7.6.3.1 Описание команды

Команда ciphers выводит на экран список поддерживаемых криптонаборов протокола SSL/TLS.

7.6.3.2 Формат ввода команды

openssl ciphers [-v] [-V] [-ssl3] [-tls1] [filter]

7.6.3.3 Опции команды

Опция	**Описание**
-v	Выводить подробную информацию с поддерживаемых криптонаборах, включающую версию протокола SSL/TLS, алгоритм обмена ключами, алгоритм аутентификации, алгоритм шифрования и алгоритм имитозащиты.
-V	Аналогично –v, но дополнительно выдаются шестнадцатеричные коды криптонаборов.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-ssl3, -tls1	Выводит на экран список шифров, совместимых с SSLv3, TLSv1, TLSv1.1 или TLSv1.2. Поскольку криптонаборы для других версий протокола SSL/TLS в настоящее время не поддерживаются, эти опции никак не влияют на результат работы программы.
-h, -?	Выводит краткое справочное сообщение.
filter	Выводить не все поддерживаемые криптонаборы, а талько соответвующие указанному фильтру.

7.6.3.4 Фильтры

Применительно к криптонаборам, использующим алгоритмы ГОСТ, могут быть полезны следующие фильтры:

Опция	Описание
eNULL, NULL	NULL-шифры – это те, которые не предполагают шифрования.
aGOST	Шифр-сьюты, использующие ГОСТ Р 34.10 (2012 или 2001) для аутентификации.
aGOST12	Шифр-сьюты, использующие аутентификацию по ГОСТ Р 34.10-2012.
kGOST	Шифр-сьюты, использующие обмен ключами VKO 34.10.
GOST12	Шифр-сьюты, использующие HMAC на базе ГОСТ Р 34.11-2012.
GOST94	Шифр-сьюты, использующие HMAC на базе ГОСТ Р 34.11-94.
GOST89MAC	Шифр-сьюты, использующие ГОСТ 28147-89 MAC вместо HMAC.

7.6.3.5 Имена критонаборов, использующих алгоритмы ГОСТ

```
TLS_GOSTR341112_256_WITH_28147_CNT_IMIT GOST2012-GOST8912-GOST8912
TLS_GOSTR341001_WITH_28147_CNT_IMIT     GOST2001-GOST89-GOST89
TLS_GOSTR341112_256_WITH_NULL_GOSTR3411 GOST2012-NULL-GOST12
TLS_GOSTR341001_WITH_NULL_GOSTR3411     GOST2001-NULL-GOST94
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.3.6 Примеры

Вывести подробный список всех криптонаборов, использующих аутентификацию по ГОСТ:

```
openssl ciphers -v aGOST
```

Вывести подробный список всех криптонаборов, использующих аутентификацию по ГОСТ, кроме наборов без шифрования:

```
openssl ciphers -v 'aGOST:!NULL'
```

7.6.4 Команда `cms`

7.6.4.1 Описание команды

Команда cms (Cryptographic message syntax) обрабатывает почтовые сообщения в формате S/MIME в.3.1. Команда может зашифровывать, расшифровывать, подписывать и проверять, сжимать и разжимать такие сообщения.

7.6.4.2 Формат ввода команды

openssl cms [-encrypt] [-decrypt] [-sign] [-verify] [-cmsout] [-resign] [-data_create] [-data_out] [-digest_create] [-digest_verify] [-compress] [-uncompress] [-EncryptedData_encrypt] [-sign_receipt] [-verify_receipt receipt] [-in filename] [-inform SMIME|PEM|DER] [-rctform SMIME|PEM|DER] [-out filename] [-outform SMIME|PEM|DER] [-stream [-indef] [-noindef] [-content filename] [-text] [-noout] [-print] [-CAfile file] [-CApath dir] [-md digest] [-[cipher]] [-nointern] [-no_signer_cert_verify] [-nocerts] [-noattr] [-nosmimecap] [-binary] [-nodetach] [-certfile file] [-certsout file] [-signer file] [-recip file] [-keyid] [-receipt_request_all] [-receipt_request_first] [-receipt_request_from emailaddress] [-receipt_request_to emailaddress] [-receipt_request_print] [-secretkey key] [-secretkeyid id] [-econtent_type type] [-inkey file] [-passin arg] [-rand file(s)] [cert.pem...] [-to addr] [-from addr] [-subject subj] [cert.pem]...

7.6.4.3 Опции команды

Существуют четырнадцать операционных опций, определяющих тип выполняемой операции. Значение остальных опций изменяется в соответствии с типом операции.

Опция	**Описание**
-encrypt	Зашифровывает почту для указанных сертификатов получателей. Входной файл — зашифровываемое сообщение. Выходной файл — зашифрованное сообщение в формате MIME. Реальный тип CMS — EnvelopedData.
-decrypt	Расшифровывает почту с использованием предоставленного сертификата и закрытого ключа. Ожидает зашифрованное почтовое сообщение в формате MIME в качестве входного файла. Расшифрованное сообщение записывается в выходной файл.

Опция	Описание
-sign	Подписывает почту с использованием предоставленного сертификата и закрытого ключа. Входной файл — подписываемое сообщение. Подписанное сообщение в формате MIME записывается в выходной файл.
-verify	Проверяет подпись под сообщением. Ожидает подписанное почтовое сообщение в качестве входного файла и выводит данные о подписи. Поддерживаются как режим открытого текста, так и непрозрачный режим.
-cmsout	В качестве входного файла принимает сообщение и выводит PEM-закодированную CMS-структуру.
-resign	Добавляет к сообщению одну или более новых подписей.
-data_create	Создает CMS типа data.
-data_out	Получает в качестве входного файла CMS типа data и выводит его содержимое.
-digest_create	Создает CMS типа DigestedData.
-digest_verify	Проверяет CMS типа DigestedData и выводит его содержимое.
-compress	Создает CMS типа CompressedData. Чтобы данная опция работала, OpenSSL должна быть скомпилирована с поддержкой zlib, иначе программа выведет ошибку.
-uncompress	Разжимает CMS типа CompressedData и выводит ее содержимое. Чтобы данная опция работала, OpenSSL должна быть скомпилирована с поддержкой zlib, иначе программа выведет ошибку.
-EncryptedData_encrypt	Зашифровывает предоставленное содержимое использованием предоставленного симметричного ключа и алгоритма с использованием CMS типа EncryptedData и выводит содержимое.
-sign_receipt	Создает и выводит подписанную квитанцию на предоставленное сообщение. Входное сообщение должно содержать запрос на подписанную квитанцию. Во всех остальных отношениях данная команда функционально эквивалентна команде -sign.

Опция	Описание
-verify_receipt receipt	Проверяет подписанную квитанцию из файла с именем, определенным параметром receipt. Входное сообщение должно содержать исходный запрос на квитанцию. Во всех остальных отношениях данная команда функционально эквивалентна команде -verify.
-in filename	Входное сообщение, которое необходимо зашифровать или подписать, или сообщение, которое необходимо расшифровать или проверить.
-inform SMIME|PEM|DER	Определяет входной формат для структуры CMS. Значение по умолчанию — SMIME, в этом случае команда ожидает сообщение в формате S/MIME. Если указано значение PEM или DER, команда ожидает CMS-структуру формата PEM или DER. Сейчас это влияет только на входной формат CMS-структуры, если никакой CMS-структуры не вводится (например, с опциями -encrypt или -sign), эта опция не оказывает никакого влияния.
-rctform SMIME|PEM|DER	Определяет формат подписанной квитанции для использования с операцией -receipt_verify.
-out filename	Текст сообщения, которое было расшифровано или проверено, или выходное сообщение формата MIME, которое было подписано или проверено.
-outform SMIME|PEM|DER	Определяет выходной формат CMS-структуры. Значение по умолчанию — SMIME, в этом случае команда выводит сообщение в формате S/MIME. Если указано значение PEM или DER, команда выводит CMS-структуру формата PEM или DER. Сейчас это влияет только на выходной формат CMS-структуры, если никакой CMS-структуры не выводится (например, с опциями -encrypt или -sign), эта опция не оказывает никакого влияния.
-stream -indef	Опции -stream и -indef эквивалентны и предоставляют возможность потокового ввода/вывода для операций кодирования. Это позволяет обрабатывать данные в один проход без необходимости держать все содержимое в памяти, что в потенциале дает возможность обрабатывать очень большие файлы. Потоковый ввод/-вывод автоматически включается для подписания сообщений в формате S/MIME с отделенными данными, если выходной формат — SMIME, в настоящее время для всех остальных операций по умолчанию потоковый ввод/вывод отключен.

Опция	Описание
-noindef	Отключает потоковый ввод/вывод там, где он создал бы сложное кодирование неопределенной длины (ASN.1 constructed encoding). В настоящее время данная опция не работает. В будущем потоковый ввод/вывод будет включаться по умолчанию во всех соответствующих операциях, и эта опция будет его отключать.
-content filename	Указывает файл, содержащий отделенные данные, осмысленно использовать только вместе с командой -verify. Полезна только в том случае, если CMS-структура использует форму отделенной подписи, в которую не включено содержание сообщения. Эта опция перезапишет любое содержимое, если входной формат S/MIME, и она использует тип содержимого multipart/signed MIME.
-text	Эта опция добавляет заголовки MIME, соответствующие типу сообщения text/plain, к предоставленному сообщению при зашифровании или подписывании. При расшифровании или проверке подписи она отбрасывает текстовые заголовки: если расшифрованное или проверенное сообщение не относится к MIME-типу text/plain, программа выдает ошибку.
-noout	Для операции -cmsout не выводит анализируемую CMS-структуру. Это полезно в сочетании с опцией -print или при проверке синтаксиса CMS-структуры.
-print	Для операции -cmsout выводит все поля CMS-структуры. Это полезно в основном для тестовых целей.
-CAfile file	Файл, содержащий доверенные сертификаты удостоверяющих центров, используется только с опцией -verify.
-CApath dir	Каталог, содержащий доверенные сертификаты удостоверяющих центров, используется только с опцией -verify. Этот каталог должен быть стандартным каталогом сертификатов, то есть хэш-сумма каждого значения поля subject name (используется команда x509 с опцией -hash) должна быть связана с каждым сертификатом.
-md digest	Алгоритм дайджеста для использования при подписывании или переподписывании. Если эта опция не указана, то будет использован умолчательный алгоритм дайджеста для ключа подписи (обычно SHA1).

Опция	Описание
-[cipher]	Алгоритм зашифрования, который следует использовать. Например, тройной DES (168-битный) — -des3 или 256-битный AES — -aes256. Может быть использовано любое стандартное наименование алгоритма (используемое в функции the EVP_get_cipherbyname()) со знаком «-» впереди, напрмер -aes_128_cbc. См. в enc список алгоритмов зашифрования, поддерживаемых вашей версией OpenSSL. Если эта опция не указана, используется тройной DES. Используется только с опциями -encrypt и -EncryptedData_create.
-nointern	При проверке подписи, как правило, сертификат подписи ищется среди сертификатов, включенных в сообщение (если таковые есть). При указании этой опции сертификат ищется только среди сертификатов, указанных в опции -certfile. Однако сертификаты в сообщении могут быть использованы как недоверенные сертификаты удостоверяющих центров.
-no_signer_cert_verify	При проверке подписанного сообщения не проверять сертификат подписи.
-nocerts	Как правило, при подписывании сообщения сертификат подписи включается в сообщение. С данной опцией включение сертификата подписи в сообщение исключается. Это уменьшит объем подписанного сообщения, но адресат должен иметь копию сертификата подписи отправителя на своем компьютере (например, переданную при помощи опции -certfile).
-noattr	Как правило, когда сообщение подписано, в него включается набор атрибутов, включающий время подписания и поддерживаемые симметричные алгоритмы. При указании этой опции такие атрибуты не включаются в сообщение.
-nosmimecap	Исключает список поддерживаемых алгоритмов атрибутов подписи; другие опции, в частности, время подписания и тип содержания, по-прежнему включаются.
-binary	Как правило, входное сообщение конвертируется в «канонический» формат, который использует CR и LF в качестве знака перевода строки, как требуется по спецификации S/MIME. При указании данной опции никакой переконвертации не происходит. Это полезно при передаче бинарных данных, которые не могут быть в MIME-формате.

Опция	Описание
-nodetach	При подписывании сообщения используется непрозрачный режим. Эта форма более устойчива к передаче по почтовым каналам, но ее не могут прочесть почтовые клиенты, которые не поддерживают S/MIME. Если эта опция не указана, используется открытый режим подписи с multipart/signed MIME-типом.
-certfile file	Позволяет указать дополнительные сертификаты. При подписывании сообщения указанные сертификаты будут включены в сообщение. При проверке подписи среди них будут искаться сертификаты подписи. Сертификаты должны быть в формате PEM.
-certsout file	Все сертификаты, содержащиеся в сообщении, записываются в файл.
-signer file	Сертификат подписи при подписывании или переподписывании сообщения. Эта опция может использоваться многократно, если требуется более одной подписи. Если сообщение проверяется, то при успешной проверке сертификаты подписи будут записаны в этот файл.
-recip file	Сертификат получателя при расшифровании сообщения. Этот сертификат должен соответствовать одному из получателей сообщения, в противном случае программа выдает ошибку.
-keyid	Использует для идентификации сертификатов расширение сертификата subject key identifier вместо имени удостоверяющего центра и серийного номера. Предоставленный сертификат должен включать расширение subject key identifier. Поддерживается опциями -sign и -encrypt.
-receipt_request_all -receipt_request_first	Для опции -sign включает запрос на подписанную квитанцию. Указывает, что запросы должны быть предоставлены всеми получателями или получателями первого уровня (непосредственными адресатами, а не теми, кто получает сообщение через список рассылки). Игнорируется, если указана опция -receipt_request_from.
-receipt_request_from emailaddress	Для опции -sign включает запрос на подписанную квитанцию. Требует прямого указания электронного адреса, который должен предоставить квитанции.
-receipt_request_to emailaddress	Добавляет прямое указание электронного адреса, на который должны быть отправлены подписанные квитанции. Эта опция должна быть указана при запрашивании подписанной квитанции.

Опция	Описание
-receipt_request_print	Для опции -verify выводит содержимое любых запросов на подписанные квитанции.
-secretkey key	Указывает симметричный ключ, который следует указать. Этот ключ должен быть предоставлен в шестнадцатеричном виде и соответствовать используемому алгоритму. Поддерживается опциями -EncryptedData_encrypt, -EncrryptedData_decrypt, -encrypt and -decrypt. Когда используется с опциями -encrypt или -decrypt, предоставленный ключ используется для сворачивания или разворачивания ключа, на котором зашифровано содержимое, с использованием ключа AES в типе KEKRecipientInfo.
-secretkeyid id	Идентификатор ключа для предоставленного симметричного ключа для типа KEKRecipientInfo. Эта опция должна присутствовать, если с опцией -encrypt используется опция -secretkey. С опцией -decrypt идентификатор используется для нахождения соответствующего ключа; если он не предоставлен, тогда происходит попытка расшифрования всех структур типа KEKRecipientInfo.
-econtent_type type	устанавливает энкапсулированный тип содержания в тип, указанный параметром type; если не предоставлен, используется тип Data. Параметр type может быть любым валидным именем OID как в текстовом, так и в числовом формате.
-inkey file	Закрытый ключ для использования при подписывании или расшифровании. Ключ должен подходить для соответствующего сертификата. Если эта опция не указана, закрытый ключ должен быть включен в файл сертификата, указанный в параметре file опций -recip или -signer. При подписывании эта опция может быть использована неоднократно для указания последовательных ключей.
-keyopt name:opt	Для подписи и шифрования эта опция может использоваться многократно, чтобы применять пользовательские параметры для ключа или сертификата, указанного предыдущей опцией. Она может использоваться для установки RSA-PSS для подписи, RSA-OAEP для шифрования или для изменения параметров по умолчанию для ECDH.
-passin arg	Источник пароля к закрытому ключу. За дополнительной информацией о формате аргумента см. раздел 7.4.

Опция	Описание
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
cert.pem...	Один или более сертификатов получателей сообщения. Используется при зашифровании сообщения.
-to, -from, -subject	Соответствующие заголовки сообщения. Они включаются в зашифрованное сообщение в незашифрованном виде, так что они могут быть включены вручную. При подписывании многие почтовые клиенты, поддерживающие S/MIME, проверяют, соответствует ли электронный адрес, указанный в сертификате подписи, электронному адресу, указанному в поле From: address.
-purpose, -ignore_critical, -issuer_checks, -crl_check, -crl_check_all, -policy_check, -extended_crl, -x509_strict, -policy, -verify_depth, -check_ss_sig, -use_deltas	Устанавливает различные опции проверки цепочки доверия. См. раздел 7.6.26.

7.6.4.4 Примечания

Сообщение MIME-типа должно быть отправлено без пустых строк между заголовками и телом сообщения. Некоторые почтовые программы автоматически добавляют пустую строку. Передача сообщения прямо в команду sendmail через конвейер — один из способов достичь корректного формата.

Подписываемое или зашифровываемое сообщение должно включать необходимоые MIME-заголовки, или многие S/MIME-клиенты покажут его некорректно (если вообще покажут). Вы можете использовать опцию -text для автоматического добавления текстовых заголовков.

«Подписанное и зашифрованное» сообщение — сообщение, которое сначала было подписано, а потом зашифровано. Этого можно добиться, зашифровывая уже подписанное сообщение (см. раздел 7.6.4.7).

Данная версия программы дает возможность подписывания каждого сообщения только одной подписью, но она проверяет несколько подписей в полученных сообщениях. Некоторые S/MIME-клиенты «падают», если сообщение содержит несколько подписей. Можно подписывать сообщения «параллельно», подписывая уже подписанное сообщение.

Опции -encrypt и -decrypt отражают обычное использование в S/MIME-клиентах. Строго говоря, они обрабатывают CMS-упакованные данные. Для других целей используются CMS-зашифрованные данные.

Опция -resign использует дайджест существующего сообщения при добавлении новой подписи. Это означает, что хотя бы в одной существующей подписи должны присутствовать атрибуты, использующие тот же дайджест сообщения, или операция не будет выполнена.

Опции -stream и -indef обеспечивают экспериментальную поддержку потокового ввода/-вывода. Результат кодируется в кодировку BER с использованием сложного кодирования с неопределенной длиной, а не в кодировку DER. Подоковый ввод/вывод поддерживается для опции -encrypt, а также для опции -sign, если содержание не отделено от подписи.

Потоковый ввод/вывод всегда используется для опции -sign с отделенными данными, но поскольку содержимое не является частью CMS-структуры, оно остается в кодировке DER.

7.6.4.5 Коды выхода

0	Операция выполнена полностью
1	Произошла ошибка при анализе опций команды
2	Один из входных файлов не может быть прочитан
3	Ошибка при создании CMS-файла или чтении MIME-сообщения.
4	Ошибка при расшифровке или проверке сообщения
5	Сообщение было проверено, но произошла ошибка при записи сертификатов подписи.

7.6.4.6 Совместимость с форматом PKCS#7

Утилита smime может обрабатывать только старый формат PKCS#7. Утилита cms поддерживает формат Cryptographic Message Syntax. Использование некоторых возможностей дает сообщения, которые не могут быть обработаны приложениями, поддерживающими только старый формат. Они указаны ниже.

Использование опции -keyid с опциями -sign или -encrypt.

Опция -outform PEM использует различные заголовки.

Опция -compress.

Использование опции -secretkey с опцией -encrypt.

Кроме того, опции -EncryptedData_create и -data_create type не могут быть обработаны старой командой smime.

7.6.4.7 Примеры

Создать сообщение, подписанное в режиме открытого текста:

```
openssl cms -sign -in message.txt -text -out mail.msg
-signer mycert.pem
```

Создать сообщение, подписанное в непрозрачном режиме:

```
openssl cms -sign -in message.txt -text -out mail.msg
-nodetach -signer mycert.pem
```

Создать подписанное сообщение, включить несколько дополнительных сертификатов и прочитать закрытый ключ из другого файла:

```
openssl cms -sign -in in.txt -text -out mail.msg
-signer mycert.pem -inkey mykey.pem -certfile mycerts.pem
```

Создать подписанное сообщение с двумя подписями, используя расширение key identifier:

```
openssl cms -sign -in message.txt -text -out mail.msg
-signer mycert.pem -signer othercert.pem -keyid
```

Отправить подписанное сообщение в ОС Unix, прямо в sendmail, включая заголовки:

```
openssl cms -sign -in in.txt -text -signer mycert.pem
-from steve@openssl.org -to someone@somewhere
-subject "Signed message" | sendmail someone@somewhere
```

Проверить сообщение, в случае успешной проверки извлечь сертификат подписи:

```
openssl cms -verify -in mail.msg -signer user.pem -out signedtext.txt
```

Отправить зашифрованное сообщение с использованием тройного алгоритма DEs:

```
openssl cms -encrypt -in in.txt -from steve@openssl.org
-to someone@somewhere -subject "Encrypted message"
-des3 user.pem -out mail.msg
```

Подписать и зашифровать сообщение:

```
openssl cms -sign -in ml.txt -signer my.pem -text
openssl cms -encrypt -out mail.msg
-from steve@openssl.org -to someone@somewhere
-subject "Signed and Encrypted message" -des3 user.pem
```

Примечание: команда зашифрования не включает опцию -text, потому что зашифровываемое сообщение уже включает MIME-заголовки.

Расшифровать сообщение:

```
openssl cms -decrypt -in mail.msg -recip mycert.pem -inkey key.pem
```

Вывод функции подписывания веб-форм в Netscape является PKCS#7-структурой в формате с отделенной подписью. Вы можете использовать эту программу для проверки подписи, свернув (line wrap) структуру, закодированную в base64, и окружив ее заголовками:

—-BEGIN PKCS7—-

—-END PKCS7—-

и воспользовавшись командой

```
openssl cms -verify -inform PEM -in signature.pem -content content.txt
```

Или же вы можете декодировать подпись из кодировки base64 и использовать

```
openssl cms -verify -inform DER -in signature.der -content content.txt
```

Создать зашифрованное сообщения, используя 128-битный алгоритм Camellia:

```
openssl cms -encrypt -in plain.txt -camellia128 -out mail.msg cert.pem
```

Добавить подпись к существующему сообщению:

```
openssl cms -resign -in mail.msg -signer newsign.pem -out mail2.msg
```

7.6.5 Команда `crl`

7.6.5.1 Описание команды

Команда crl обрабатывает файлы списков отзыва сертификатов в формате DER или PEM.

7.6.5.2 Формат ввода команды

openssl crl [-inform PEM|DER] [-outform PEM|DER][-text] [-in filename] [-out filename] [-noout] [-hash] [-issuer] [-lastupdate] [-nextupdate] [-CAfile file] [-CApath dir] [-fingerprint] [crlnumber] [-nameopt options]

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.5.3 Опции команды

Опция	**Описание**
-inform DER|PEM	Указывает входной формат. Формат DER — структура CRL в DER-кодировке. PEM (умолчание) — версия DER-формы в кодировке base64 с верхним и нижним ограничителями.
-outform DER|PEM	Указывает выходной формат. Опция имеет те же значения, что и опция -inform.
-in filename	Указывает файл с входными данными. Если опция не указана, данные читаются со стандартного ввода.
-out filename	Указывает файл для записи выходных данных. Если опция не указана, данные выводятся в стандартный вывод.
-text	Вывести список отзыва сертификатов в текстовом виде.
-noout	Не выводить кодированную версию списка отзыва сертификатов.
-hash	Вывести хэш-сумму значения поля issuer. Эту опцию можно использовать для поиска списков отзыва сертификатов в каталоге по полю issuer name.
-hash_old	Выводит хэш-значение имени издателя списка отозванных сертификатов, используя устаревший алгоритм, который использовался версиями OpenSSL до версии 1.0.0.
-issuer	Вывести имя выпускающего.
-lastupdate	Вывести значение поля lastUpdate.
-nextupdate	Вывести значение поля nextUpdate.
-CAfile file	Проверить подпись под списком отзыва сертификатов с поиском сертификата выпустившего удостоверяющего центра в файле
-CApath dir	Проверить подпись под списком отзыва сертификатов с поиском сертификата выпустившего удостоверяющего центра в каталоге. Этот каталог должен быть стандартным каталогом сертификатов, то есть с каждым сертификатом должен быть связан хэш значения соответствующего поля subject name (полученного с помощью утилиты x509 с опцией -hash).

7.6.5.4 Примечания

PEM-формат списка отзыва сертификатов использует следующие верхний и нижний ограничители:

```
-----BEGIN X509 CRL-----
-----END X509 CRL-----
```

7.6.5.5 Примеры

Преобразовать файл списка отзыва сертификатов из формата PEM в формат DER:

```
openssl crl -in crl.pem -outform DER -out crl.der
```

Output the text form of a DER encoded certificate:

```
openssl crl -in crl.der -text -noout
```

7.6.6 Команда `crl2pkcs7`

7.6.6.1 Описание команды

Команда crl2pkcs7 превращает один или более сертификатов и список отзыва сертификатов (необязателен) в вырожденную PKCS#7-структуру, содержащую только сертификаты (не содержащую сообщений).

7.6.6.2 Формат ввода команды

openssl crl2pkcs7 [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-out filename] [-certfile filename] [-nocrl]

7.6.6.3 Опции команды

Опция	**Описание**
-inform DER|PEM	Указывает входной формат списка отзыва сертификатов. Формат DER — структура CRL в DER-кодировке. PEM (умолчание) — версия DER-формы в кодировке base64 с верхним и нижним ограничителями.
-outform DER|PEM	Указывает выходной формат PKCS#7-структуры. Формат DER — структура CRL в DER-кодировке. PEM (умолчание) — версия DER-формы в кодировке base64 с верхним и нижним ограничителями.
-in filename	Эта опция определяет входной файл, из которого следует считать список отзыва сертификатов. Если эта опция не указана, список отзыва сертификатов считывается со стандартного ввода.
-out filename	Эта опция определяет выходной файл, в который записывается полученная PKCS#7-структура. Если эта опция не указана, PKCS#7-структура выводится на стандартный вывод.
-certfile filename	Эта опция определяет файл, содержащий один или больше сертификатов в PEM-формате. Все сертификаты из этого файла будут включены в PKCS#7-структуру. Эта опция может быть указана несколько раз, если нужно считать сертификаты из нескольких файлов.
-nocrl	Как правило, в выходной файл включается список отзыва сертификатов. Если указана данная опция, список отзыва сертификатов не считывается из входных данных и не включается в выходной файл.

7.6.6.4 Примеры

Создать PKCS#7-структуру из сертификата и списка отзыва сертификатов:

```
openssl crl2pkcs7 -in crl.pem -certfile cert.pem -out p7.pem
```

Создать PKCS#7-структуру в DER-формате из нескольких разных сертификатов, список отзыва сертификатов не включать:

```
openssl crl2pkcs7 -nocrl -certfile newcert.pem -certfile
demoCA/cacert.pem -outform DER -out p7.der
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.6.5 Примечания

Выходной файл представляет собой PKCS#7-структуру «signed data», не содержащую никаких подписей, содержащую только сертификаты и опциональный список отзыва сертификатов.

Эту утилиту можно использовать для отправки сертификатов и списков отзыва сертификатов в браузеры в качестве части процесса ввода сертификата в действие. Это включает отправку данных MIME-типа application/x-x509-user-cert.

Данные в PEM-формате без верхнего и нижнего ограничителей можно использовать для установки пользовательских сертификатов и сертификатов УЦ в Microsoft Internet Explorer с помощью Active-X элемента Xenroll.

7.6.7 Команда `dgst`

7.6.7.1 Описание команды

Команда позволяет вычислить хэш-сумму для предоставленного файла или файлов в шестнадцатеричном виде. Также может быть использована для формирования и подтверждения электронной подписи (ЭП). При работе с алгоритмами ГОСТ всегда необходимо указывать опцию -md_gost12_256, -md_gost12_512 или -md_gost94 (последнюю только для обепечения совместимости с предыдущими версиями).

7.6.7.2 Формат ввода команды

openssl dgst [-md_gost12_256|-md_gost12_512|-md_gost94|-sha|-sha1|-mdc2|-ripemd160|-sha224|-sha256|-sha384|-sha512|-md2|-md4|-md5|-dss1] [-c] [-d] [-hex] [-binary] [-r] [-non-fips-allow] [-out filename] [-sign filename] [-keyform arg] [-passin arg] [-verify filename] [-prverify filename] [-signature filename] [-hmac key] [-non-fips-allow] [-fips-fingerprint] [file...]

7.6.7.3 Опции команды

Опция	**Описание**
-md_gost12_256	Использовать алгоритм дайджеста ГОСТ Р 34.11-2012, 256 бит
-md_gost12_512	Использовать алгоритм дайджеста ГОСТ Р 34.11-2012, 512 бит
-md_gost94	Использовать алгоритм дайджеста ГОСТ Р 34.11-94
-c	Вывести хэш-сумму в двух цифровых группах, разделенных вертикальной чертой, актуальна только если используется шестнадцатеричный формат вывода.
-d	Вывести отладочную информацию BIO.
-hex	Вывести хэш-сумму в виде шестнадцатеричного дампа. Это умолчательный вариант для «обыкновенной» хэш-суммы, в отличие от цифровой подписи. См. раздел Примечания.
-binary	Вывести хэш-сумму или подпись в бинарном виде.
-r	Выводит хэш-значение в формате "coreutils используемом программами типа sha1sum.

Опция	Описание
-non-fips-allow	Позволяет использовать не-FIPS алгоритмы хэширования в режиме FIPS. Опция не имеет никакого эффекта для не-FIPS режима.
-out filename	Указать имя файла вывода, по умолчанию — стандартный вывод.
-sign filename	Выработать цифровую подпись, используя закрытый ключ, содержащийся в указанном файле.
-keyform arg	Определяет формат ключа, которым следует подписывать дайджест. Команда dgst поддерживает только форматы PEM и ENGINE.
-engine id	Использовать энджин с именем id для выполнения операции (включая хранилище закрытого ключа). Этот энджин не используется как источник алгоритмов хэширования, если только он не указазан в конфигурационном файле.
-sigopt nm:v	Передать опции в алгоритм подписи. Набор опций определяется используемым алгоритмом подписи.
-passin arg	Место хранения пароля к закрытому ключу. За дополнительной информацией о формате аргумента см. раздел 7.4.
-verify filename	Проверить подпись с использованием открытого ключа, содержащегося в указанном файле. Результат — либо «Подпись корректна», либо «Подпись некорректна».
-prverify filename	Проверить подпись с использованием открытого ключа, содержащегося в указанном файле.
-signature filename	Указать файл, под которым необходимо проверить подпись.
-hmac key	Создать имитовставку на базе хэш-функции с использованием ключа key.
-mac alg	Создать имитовставку. Самый популярный алгоритм имитовставки - HMAC (имитовставка на базе хэш-функции), но существуют другие алгоритмы MAC, которые не основываются на хэш-функции, например алгоритм gost-mac, поддерживаемый модулем engine cryptocom. Ключи и другие параметры для MAC следует устанавливать с помощью параметра -macopt.
-macopt nm:v	Передать опции в алгоритм имитозащиты, определенный ключом -mac. Следующие опции поддерживаются алгоритмами HMAC и gost-mac:
key:string	Определяет ключ алгоритма имитозащиты в виде буквенно-цифровой строки (применяется, если ключ содержит только выводимые символы). Длина строки должна соответствовать любым ограничениям алгоритма имитовставки, например, для алгоритма gost-mac она должна быть ровно 32 символа.
hexkey:string	Определяет ключ алгоритма имитозащиты в шестнадцатеричном виде (две шестнадцатеричные цифры на байт). Длина ключа должна соответствовать любым ограничениям алгоритма имитовставки, например, для алгоритма gost-mac она должна быть ровно 32 символа.

Опция	Описание
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-non-fips-allow	Включать использование не-FIPS алгоритмов, таких как MD5, даже в режиме FIPS.
-fips-fingerprint	Вычислять HMAC, используя специфичный ключ для некоторых OpenSSL-FIPS операций.
file...	Файл или файлы, для которых нужно вычислить хэш-суммы. Если не указано ни одного файла, используется стандартный ввод.

7.6.7.4 Примеры

Подсчитать хэш-значение файла в шестнадцатеричном виде:

```
openssl dgst -md_gost12_256 -hex file.txt
```

Подписать файл, используя алгоритм хэширвоания ГОСТ Р 34.11-2012 256 бит с выводом в файл в двоичном виде:

```
openssl dgst -gost12_256 -sign privatekey.pem -out signature.sign file.txt
```

Проверить подпись:

```
openssl dgst -gost12_256 -verify publickey.pem
-signature signature.sign  file.txt
```

Вычислить хэш по алгоритму ГОСТ Р 34.11-2012: - 512 бит:

```
openssl dgst -md_gost12_512 openssl.cnf
```

- 256 бит:

```
openssl dgst -md_gost12_256 openssl.cnf
```

Вычислить MAC по gost89:

```
openssl dgst -mac gost-mac -macopt
key:12345678901234567890123456789012 dgst.dat
```

Вычислить MAC по gost89 с параметрами ТК26 (набор Z):

```
openssl dgst -mac gost-mac-12 -macopt
key:12345678901234567890123456789012 dgst8.dat
```

7.6.8 Команда `errstr`

7.6.8.1 Описание команды

Иногда приложение не загружает сообщение об ошибке, и доступны только численные формы таких сообщений. Можно использовать команду errstr для вывода значения шестнадцатиричного кода. Шестнадцатиричный код — это шестнадцатиричные цифры после второй точки с запятой.

7.6.8.2 Формат ввода команды

openssl errstr error_code

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.8.3 Пример

Код ошибки:

```
27594:error:2006D080:lib(32):func(109):reason(128):bss_file.c:107:
```

может быть выведен с помощью:

```
openssl errstr 2006D080
```

которая создает сообщение об ошибке:

```
error:2006D080:BIO routines:BIO_new_file:no such file
```

7.6.9 Команда `genpkey`

7.6.9.1 Описание команды

Команда genpkey создает закрытый ключ.

7.6.9.2 Формат ввода команды

openssl genpkey [-out filename] [-outform PEM|DER|ENGINE] [-pass arg] [-cipher] [-engine id] [-paramfile file] [-algorithm alg] [-pkeyopt opt:value] [-genparam] [-text]

7.6.9.3 Опции команды

Опция	**Описание**
-out filename	Имя выходного файла. Если этот параметр не указан, используется стандартный вывод. Если указан outform ENGINE, то в этом параметре передается строка-идентификатор ключа на аппаратном ключевом носителе (см. 7.3).
-outform DER|PEM|ENGINE	Указывает выходной формат DER или PEM. Если указано значение ENGINE, то ключ будет записан на аппаратный ключевой носитель, определяемый параметром -out (см. 7.3).
-pass arg	Источник пароля для закрытого ключа. Для получения дополнительной информации о формате аргумента arg см. раздел 7.4.
-cipher	Эта опция зашифровывает закрытый ключ с помощью указанного алгоритма. Можно указывать любое название алгоритма, которое принимает функция EVP_get_cipherbyname(), например des3.
-engine id	Указание модуля engine (по его уникальной идентификационной строке) заставит команду genpkey попытаться получить функциональную ссылку на указанный модуль, инициализируя его, если необходимо. Затем модуль engine будет установлен как умолчательный для всех доступных алгоритмов. Если эта опция используется, она должна указываться перед всеми остальными опциями.

Опция	Описание
-algorithm alg	алгоритм открытого ключа, который следует использовать, такой как RSA, DSA or DH. Если эта опция используется, она должна указываться перед всеми опциями -pkeyopt. Опции -paramfile и -algorighm взаимно исключают друг друга.
-pkeyopt opt:value	Устанавливает опцию opt алгоритма открытого ключа в значение value. Точный набор поддерживаемых опций зависит от используемого алгоритма открытого ключа и его реализации. См. раздел 7.6.9.4.
-genparam	генерирует набор параметров вместо закрытого ключа. Если эта опция используется, она должна предшествовать опциям -algorithm, -paramfile или -pkeyopt.
-paramfile filename	Некоторые алгоритмы открытых ключей создают закрытый ключ на основе набора параметров. Они могут поддерживаться с помощью этой опции. Если эта опция используется, используемый алгоритм открытого ключа определяется параметрами. Эта опция должна указываться перед опциями -pkeyopt. Опции -paramfile и -algorithm взаимно исключают друг друга.
-text	Выводит (в незашифрованном виде) текстовое представление закрытого и открытого ключей и параметров вместе с PEM- или DER-структурой.

7.6.9.4 Опции создания ключей ГОСТ

Опции, поддерживаемые каждым алгоритмом, а в действительности и каждой реализацией алгоритма, могут отличаться. В данном разделе указаны опции для создания ключей ГОСТ в реализации OpenSSL.

Поддержка ГОСТ Р 34.10 по умолчанию не включена. Чтобы включить поддержку этого алгоритма, следует загрузить модуль cryptocom в конфигурационном файле OpenSSL.

Использование файла параметров для алгоритма ГОСТ Р 34.10 опционально. Параметры могут быть указаны как непосредственно во время создания ключей, так и во время создания файла параметров.

paramset:name

Указывает набор параметров ГОСТ Р 34.10-2012 512 бит в соответствии с методическими рекомендациями ТК26 либо набор параметров ГОСТ Р 34.10-2012 256 бит и ГОСТ Р 34.10-2001 в соответствии с RFC 4357. Набор параметров может быть указан с использованием аббревиатуры, сокращенного названия или численного OID. Поддерживаются следующие наборы параметров:

Для ГОСТ Р 34.10-2012 512 бит:

Набор параметров	OID	Использование
A	1.2.643.7.1.2.1.2.1	Подпись и ключевой обмен
B	1.2.643.7.1.2.1.2.2	Подпись и ключевой обмен

Для ГОСТ Р 34.10-2012 256 бит и ГОСТ Р 34.10-2001:

Набор параметров	OID	Использование
A	1.2.643.2.2.35.1	Подпись
B	1.2.643.2.2.35.2	Подпись
C	1.2.643.2.2.35.3	Подпись
XA	1.2.643.2.2.36.0	Ключевой обмен
XB	1.2.643.2.2.36.1	Ключевой обмен
test	1.2.643.2.2.35.0	Для тестовых целей

7.6.9.5 Примечания

Использование команды genpkey считается более предпочтительным, чем использование утилит, связанных с конкретными алгоритмами, потому что с этой командой можно использовать дополнительные опции алгоритмов, а также алгоритмы из подгружаемых модулей engine.

7.6.9.6 Примеры

Создать закрытый ключ алгоритма ГОСТ Р 34.10-2012 256 бит с набором параметров A:

```
openssl genpkey -algorithm gost2012_256 -pkeyopt paramset:A -out key.pem
```

Зашифровать создаваемый закрытый ключ, используя алгоритм ГОСТ 28147-89 и пассфразу hello:

```
openssl genpkey -algorithm gost2012_256 -pkeyopt paramset:A -out key.pem
-gost89 -pass pass:hello
```

7.6.10 Команда `ocsp`

7.6.10.1 Описание команды

OCSP (онлайн-протокол статусов сертификатов) позволяет приложениям определять (отозванное) состояние идентифицированного сертификата (RFC 6960).

Команда oscp выполняет многие обычные задачи OSCP. Ее можно использовать для вывода запросов и ответов на них, создавать и посылать запросы на OCSP-ответчик, а также в качестве небольшого oscp-сервера.

7.6.10.2 Формат ввода команды

openssl ocsp [-out file] [-issuer file] [-cert file] [-serial n] [-signer file] [-signkey file] [-sign_other file] [-no_certs] [-req_text] [-resp_text] [-text] [-reqout file] [-respout file] [-reqin file] [-respin file] [-nonce] [-no_nonce] [-url URL] [-host host:n] [-path] [-CApath dir] [-CAfile file] [-no_alt_chains] [-VAfile file] [-validity_period n] [-status_age n] [-noverify] [-verify_other file] [-trust_other] [-no_intern] [-no_signature_verify] [-no_cert_verify] [-no_chain] [-no_cert_checks] [-no_explicit] [-port num] [-index file] [-CA file] [-rsigner file] [-rkey file] [-rother file] [-resp_no_certs] [-nmin n] [-ndays n] [-resp_key_id] [-nrequest n] [-md5|-sha1|...]

7.6.10.3 Опции команды

7.6.10.3.1 Клиентские опции команды oscp

Опция	**Описание**
-out filename	Указывает имя выходного файла. По умолчанию выходные данные направляются на стандартный выход.
-issuer filename	Указывает файл сертификата удостоверяющего центра, выпустившего проверяемый сертификат. Эту опцию можно использовать несколько раз. Сертификат, указанный в качестве значения опции, должен быть в формате PEM.
-cert filename	Добавляет к формируемому запросу запрос статуса сертификата, содержащегося в файле filename. Информация об удостоверяющем центре берется из предыдущей опции issuer; если ни одной такой опции не указано, выводится сообщение об ошибке.

Опция	Описание
-serial num	То же, что и опция cert, за исключением того, что к формируемому запросу добавляется запрос о статусе сертификата с серийным номером, указанным в качестве значения опции num. Серийный номер интерпретируется как десятичное целое число, если только он не начинается с 0x. Можно указывать также отрицательные целые числа с помощью знака «-» перед значением num.
-signer filename, -signkey filename	Подписывает ocsp-запрос с использованием сертификата, указанного в опции signer, и закрытого ключа, указанного в опции signkey. Если опция signkey не указана, закрытый ключ считывается из того же файла, что и сертификат. Если не указана ни одна опция, ocsp-запрос не подписывается.
-sign_other filename	Дополнительные сертификаты, которые следует включить в подписанный запрос.
-nonce, -no_nonce	Включает OCSP-расширение nonce в запрос или отключает добавление этого расширения. Как правило, если OCSP-запрос вводится с использованием опции respin, расширение nonce не включается; указывание опции nonce указывает на необходимость включения расширения nonce. Если OCSP-запрос создается (с использованием опции cert и serial), расширение nonce добавляется автоматически; указанием опции no_nonce добавление расширения nonce отменяется.
-req_text, -resp_text, -text	Выводит текстовую форму OCSP-запроса, ответа или и то и другое соответственно.
-reqout file, -respout file	Выводит запрос статуса сертификата или ответ в DER-кодировке в файл.
-reqin file, -respin file	Считывает OCSP-запрос или файл ответа из файла. Эти опции игнорируются, если другими опциями подразумевается создание OCSP-запроса или ответа (например, присутствуют опции serial, cert или host).
-url responder_url	Определяет URL ответчика. Могут указываться как URL, начинающиеся с HTTP, так и с HTTPS (SSL/TLS).
-host hostname:port, -path pathname	Если опция host присутствует, то OCSP-запрос отправляется на указанный port указанного hostname. path указывает http-путь или «/» по умолчанию.
-timeout seconds	Время ожидания соединения со службой OCSP в секундах.
-CAfile file, -CApath pathname	Файл, содержащий сертификаты доверенных удостоверяющих центров. Они используются для проверки подписи на OCSP-ответе.
-no_alt_chains	Подробное описание смотри в разделе опции verify данного руководства.
-verify_other file	Файл, содержащий дополнительные сертификаты, среди которых следует искать сертификат, на котором подписан OCSP-ответ. Некоторые ответчики удаляют из ответа сертификат подписчика; эту опцию можно использовать, чтобы в таких случаях предоставить необходимый сертификат.

Опция	Описание
-trust_other	сертификаты, указанные в опции verify_certs, должны быть явным образом указаны как доверенные, и над ними не будет производиться никаких дополнительных проверок. Это полезно в тех случаях, когда полная цепочка сертификата ответчика н едоступна или корневой сертификат не является доверенным.
-VAfile file	Файл, содержащий сертификаты ответчиков, явным образом указанные как доверенные. Эквивалент опций verify_certs и -trust_other.
-noverify	Не пытаться проверять подпись под OCSP-ответом или значения расширения nonce. Эта опция, как правило, используется только для отладки, поскольку она отключает любую проверку сертификата ответчика.
-no_intern	Игнорировать сертификаты, содержащиеся в OCSP-ответе, во время поиска сертификата, на котором подписан данный. При указании этой опции сертификат, на котором подписан данный, должен быть указан с помощью опции -verify_certs или -VAfile.
no_signature_verify	Не проверять подпись в OCSP-ответе. Поскольку эта опция позволяет принимать ответы с некорректными подписями, она, как правило, используется только в целях тестирования.
-no_cert_verify	Вообще не проверять сертификат, на котором подписан OCSP-ответ. Поскольку эта опция позволяет подписывать OCSP-ответ любым сертификатом, ее следует использовать только в целях тестирования.
-no_chain	Не использовать сертификаты в ответе в качестве дополнительных недоверенных сертификатов удостоверяющих центров.
-no_explicit	Не доверять явно корневому CA, который используется для проверки подписи OCSP.
-no_cert_checks	Не выполнять никаких дополнительных проверок сертификата, на котором подписан OCSP-ответ. Это значит — не проверять, имеет ли право указанный сертификат предоставлять необходимую информацию по статусу; в результате эту опцию следует использовать только в целях тестирования.
-validity_period nsec, -status_age age	Эти опции определяют диапазон времени в секундах, который будет выдерживаться в OCSP-ответе. Каждый ответ по статусу сертификата включает значение поля notBefore и опционально — значение поля notAfter. Текущее время должно попадать между этими двумя величинами, но интервал между ними может быть всего несколько секунд. На практике часы OCSP-ответчика и клиентов могут не быть точно синхронизированы и проверка может не удасться. Чтобы этого избежать, можно использовать опцию -validity_period для определения приемлемого диапазона ошибок в секундах. Значение этой опции по умолчанию — 5 минут. Если в ответ не включено значение поля notAfter, это означает, что новая информация по статусу уже доступна. В этом случае проверяется возраст поля notBefore, чтобы убедиться, что он не старше чем age секунд. По умолчанию эта дополнительная проверка не проводится.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-md5, -sha1, -sha256, -ripemod160.	Эта опция определяет алгоритм хэширования для использования при генерации идентификатора сертификата удостоверяющего центра. По умолчанию используется алгоритм SHA1.

7.6.10.3.2 Серверные опции команды ocsp

Опция	**Описание**
-index indexfile	indexfile — текстовый индексный файл в формате команды ca утилиты openssl (см. раздел 7.6.2), содержащий информацию о статусе сертификатов. Если указана опция index, команда ocsp работает в режиме ответчика (сервера), в противном случае — в режиме клиента. Запрос(ы), которые обрабатывает ответчик, могут либо определяться в командной строке (с помощью опций issuer и serial), либо считываться из файла (с помощью опции respin) или с помощью внешних ocsp-клиентов (если указаны порт или url). Если указана опция index, также должны присутствовать опции CA и rsigner.
-CA file	Сертификат удостоверяющего центра, соответствующий информации об отзывах в файле indexfile.
-rsigner file	Сертификат, на котором следует подписывать OCSP-ответы.
-rother file	Дополнительные сертификаты, которые следует включить в OCSP-ответ.
-resp_no_certs	Не включать никаких сертификатов в OCSP-ответ.
-resp_key_id	Идентифицировать сертификат подписчика с использованием ID ключа, по умолчанию — использовать значение поля subject. Использовать этот режим не рекомендуется, т.к. в RFC 6960 явно прописано использование нестандартного для России алгоритма хэширования.
-rkey file	Закрытый ключ для подписывания OCSP-ответов: если эта опция не присутствует, используется файл, указанный в качестве значения опции rsigner.
-port portnum	Порт, с которого следует считывать OCSP-запросы. Этот порт также может быть указан с помощью опции url.
-nrequest number	OCSP-сервер закончит работу по получении number запросов, по умолчанию — неограниченного количества.
-nmin minutes, -ndays days	Количество минут или дней, когда доступна свежая информация об отзывах: используется в поле nextUpdate. Если ни одна из этих опций не указана, поле nextUpdate опускается, что означает, что свежая информация об отзывах доступна немедленно.

7.6.10.4 Проверка OCSP-ответов

OCSP-ответы следуют правилам, указанным в RFC 6960.

Изначально определяется местоположение сертификата OCSP-ответчика и проверяется подпись под OCSP-запросом с использованием открытого ключа из сертификата ответчика.

Затем на OCSP-ответчике происходит обычная проверка сертификата с построением цепочки сертификатов в процессе. Местонахождение доверенных сертификатов, используемых для построения цепочки, может быть определено с помощью опций CAfile и CApath, или они будут отыскиваться в стандартном каталоге сертификатов OpenSSL.

Если первичная проверка не удается, процесс проверки OCSP прекращается с сообщением

об ошибке.

В противном случае сертификат выпустившего CA в запросе сравнивается с сертификатом OCSP-ответчика: если они совпадают, проверка OCSP считается успешной.

В противном случае сертификат, выпустивший сертификат OCSP-ответчика, сравнивается с сертификатом выпускающего удостоверяющего центра в запросе. Если они совпадают и в сертификате OCSP-ответчика присутствует OCSPSigning extended key usage, проверка OCSP считается успешной.

В противном случае корневой сертификат удостоверяющего центра, выпустившего сертификат OCSP-ответчика, проверяется на предмет того, является ли он доверенным для подписывания OCSP. Если да, то проверка OCSP считается успешной.

Если ни одна из этих проверок не оказывается успешной, проверка OCSP не удается.

По сути это означает, что если сертификат OCSP-ответчика является доверенным непосредственно у того удостоверяющего центра, о котором он передает информацию об отзывах (и корректно сконфигурирован), то проверка удается.

Если OCSP-ответчик является «глобальным ответчиком», который может давать информацию о многих удостоверяющих центрах и обладает собственной цепочкой сертификатов, то его корневой сертификат может быть доверенным для OCSP-подписи. Например:

```
openssl x509 -in ocspCA.pem -addtrust OCSPSigning -out trustedCA.pem
```

Или же сертификат самого ответчика может быть явным образом объявлен доверенным с помощью опции -VAfile.

7.6.10.5 Примечания

Как уже было сказано, многие из проверочных опций предназначены для тестовых и отладочных целей. Как правило, нужно использовать только опции -CApath, -CAfile и, если ответчик — глобальный VA, то -VAfile.

OCSP-сервер полезен только для тестовых и демонстрационных целей: на самом деле он не может использоваться в качестве полноценного OCSP-ответчика. Он содержит только очень простой обработчик HTTP-запросов и может обрабатывать только POST-форму OCSP-запросов. Он также обрабатывает запросы последовательно, что означает, что он не может отвечать на новые запросы, пока не обработает текущий. Текстовый формат индексного файла отзывов сертификатов также неэффективен для больших количеств данных по отзывам сертификатов.

Возможно запускать приложение ocsp в режиме ответчика через CGI-скрипт с использованием опций respin и respout.

7.6.10.6 Примеры

Создать OCSP-запрос и записать его в файл:

tt openssl ocsp -issuer issuer.pem -cert c1.pem -cert c2.pem -reqout req.der

Отправить запрос на OCSP-ответчик с URL-адресом http://ocsp.myhost.com/, сохранить ответ в файле и вывести его в текстовой форме

```
openssl ocsp -issuer issuer.pem -cert c1.pem -cert c2.pem  -url
http://ocsp.myhost.com/ -resp_text -respout resp.der
```

Прочитать OCSP-ответ и вывести в текстовой форме:

```
openssl ocsp -respin resp.der -text
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

OCSP-сервер на порте 8888 использует стандартную конфигурацию удостоверяющего центра (см. раздел 7.6.2.6) и отдельный сертификат ответчика. Все запросы и ответы выводятся в файл.

```
openssl ocsp -index demoCA/index.txt -port 8888 -rsigner rcert.pem
-CA demoCA/cacert.pem -text -out log.txt
```

Как выше, но закончить работу после обработки одного запроса:

```
openssl ocsp -index demoCA/index.txt -port 8888 -rsigner rcert.pem
-CA demoCA/cacert.pem -nrequest 1
```

Запросить информацию о статусе с использованием внутренне генерированного запроса:

```
openssl ocsp -index demoCA/index.txt -rsigner rcert.pem -CA
demoCA/cacert.pem -issuer demoCA/cacert.pem -serial 1
```

Запросить информацию о статусе с использованием запроса, прочитанного из файла, записать ответ в другой файл.

```
openssl ocsp -index demoCA/index.txt -rsigner rcert.pem -CA
demoCA/cacert.pem -reqin req.der -respout resp.der
```

7.6.11 Команда pkcs7

7.6.11.1 Описание команды

Команда pkcs7 переводит файлы формата PKCS#7 в форматы DER или PEM.

7.6.11.2 Формат ввода команды

openssl pkcs7 [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-out filename] [-print_certs] [-text] [-noout] [-engine id]

7.6.11.3 Опции команды

Опция	**Описание**
-inform DER|PEM	Указывает входной формат. Формат DER — структура формата PKCS#7 версии 1.5 в DER-кодировке. PEM (умолчание) — версия DER-формы в кодировке base64 с верхним и нижним ограничителями.
-outform DER|PEM	Указывает выходной формат. Опция имеет те же значения, что опция -inform.
-in filename	Указывает файл с входными данными. Если эта опция не указана, данные считываются со стандартного ввода.
-out filename	Указывает файл для записи выходных данных. По умолчанию данные направляются на стандартный вывод.
-print_certs	Выводит все сертификаты и списки отзыва сертификатов, содержащиеся в файле. Перед сертификатами выводятся значения полей subject и issuer в однострочном формате.
-text	Выводит информацию о сертификатах полностью, а не только значения полей subject и issuer.

Опция	Описание
-noout	Не выводить зашифрованную версию PKCS#7-структуры (или сертификаты, если указана опция -print_certs).
-engine id	Выбор энджина (по уникальной строке id) приводит к тому, что pkcs7 будет пытаться получить функциональную ссылку на выбранный энджин, таким образом инициализируя его, если это необходимо. После этого энджин устанавливается, как энджин по умолчанию, для всех доступных алгоритмов.

7.6.11.4 Примеры

Перевести PKCS#7-файл из формата PEM в формат DER:

```
openssl pkcs7 -in file.pem -outform DER -out file.der
```

Вывести все сертификаты, содержащиеся в файле:

```
openssl pkcs7 -in file.pem -print_certs -out certs.pem
```

7.6.11.5 Примечания

PEM-формат PKCS#7-структуры использует следующий вид верхнего и нижнего ограничителей:

```
-----BEGIN PKCS7-----
-----END PKCS7-----
```

Для совместимости с некоторыми удостоверяющими центрами он также поддерживает следующий вид верхнего и нижнего ограничителей:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

7.6.11.6 Ограничения

Не существует опции, позволяющей вывести все поля PKCS#7-файла.

Команда PKCS#7 поддерживает только PKCS#7 версии 1.5, описанную в RFC2315. В настоящее время она не поддерживает, например, новый CMS, описанный в RFC2630.

7.6.12 Команда `pkcs8`

7.6.12.1 Описание команды

Команда pkcs8 работает с закрытыми ключами формата PKCS#8. Она может работать с незашифрованными ключами формата PKCS#8 PrivateKeyInfo и с зашифрованными ключами формата EncryptedPrivateKeyInfo format с различными алгоритмами формата PKCS#5 (версии 1.5 и 2.0) и PKCS#12.

7.6.12.2 Формат ввода команды

openssl pkcs8 [-topk8] [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-noiter] [-nocrypt] [-nooct] [-embed] [-nsdb] [-v2 alg] [-v1 alg] [-engine id]

7.6.12.3 Опции команды

Опция	**Описание**
-topk8	Как правило, эта команда переводит закрытый ключ формата PKCS#8 в закрытый ключ традиционного формата. Данная опция указывает на обратную ситуацию: команда считывает ключ традиционного формата и выводит ключ формата PKCS#8.
-inform DER|PEM	Указывает входной формат. Если в качестве входных данных ожидается ключ формата PKCS#8, данная опция указывает на PEM- или DER-версию. В противном случае используется PEM- или DER-формат ключа традиционного формата.
-outform DER|PEM	Указывает выходной формат. Опция имеет те же значения, что и опция -inform.
-in filename	Указывает входной файл, из которого следует считать ключ. Если эта опция не указана, ключ считывается со стандартного входа. Если ключ зашифрован, будет запрошена пассфраза.
-passin arg	Источник пароля для входного файла. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4.
-out filename	Указывает выходной файл, в который следует записать ключ. Если эта опция не указывается, ключ выводится на стандартный вывод. Если указаны какие-либо опции зашифрования, будет запрошена пассфраза. Имя выходного файла не должно совпадать с именем входного файла.
-passout arg	Источник пароля для выходного файла. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4.
-nocrypt	Как правило, ключи формата PKCS#8 являются структурами формата PKCS#8 EncryptedPrivateKeyInfo, использующими соответствующий алгоритм шифрования, основанного на пароле. Эта опция указывает, что на вводе или выходе должна быть незашифрованная структура формата PrivateKeyInfo. Эта опция полностью отменяет зашифрование закрытых ключей, и ее следует использовать только при крайней необходимости. Некоторые программы используют незашифрованные закрытые ключи.

Опция	Описание
-v2 alg	Эта опция дает возможность использования алгоритмов версии 2.0. формата PKCS#5. Как правило, закрытые ключи формата PKCS#8 зашифровываются на пароле при помощи алгоритма под названием pbeWithMD5AndDES-CBC, использующим 56-битное DES-зашифрование, но это был самый мощный алгоритм шифрования, который поддерживался версией 1.5 формата PKCS#8. Данная опция указывает на использование алгоритмов версии 2.0, которые могут использовать любой алгоритм зашифрования, такой, как 168-битный тройной DES или 128-битный RC2, но пока что не все программы поддерживают версию 2.0. Если вы работаете с закрытыми ключами только в рамках криптобиблиотеки OpenSSL, это не имеет значения. Аргумент arg — алгоритм зашифрования, который следует использовать, возможными значениями являются des, des3 и rc2. Рекомендуется использовать des3.
-v1 alg	Эта опция указывает, какой алгоритм версии 1.5 формата PKCS#5 или PKCS#12 следует использовать.
-engine id	Указывает загружаемый модуль engine (по его уникальной идентификационной строке) и вызывает попытку использовать реализацию криптографических алгоритмов из этого модуля.

7.6.12.4 Примечания

Зашифрованная форма PKCS#8-файлов в формате PEM использует следующий вид верхнего и нижнего ограничителей:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
-----END ENCRYPTED PRIVATE KEY-----
```

Незашифрованная форма использует:

```
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
```

Закрытые ключи, зашифрованные алгоритмами PKCS#5 версии 2.0 с высоким значением количества итераций более надежны, чем ключи, зашифрованные алгоритмами традиционных SSLeay-совместимых форматов. Поэтому если дополнительная безопасность считается важной, следует преобразовывать ключи.

Умолчательное зашифрование всего лишь 56-битное, потому что это зашифрование поддерживают самые современные реализации PKCS#8.

Некоторые программы могут использовать PKCS#12-алгоритмы зашифрования на пароле с закрытыми ключами формата PKCS#8: они обрабатываются автоматически, но не существует опции для их получения.

Можно записать DER-закодированные зашифрованные закрытые ключи в формате PKCS#8, потому что информация о зашифровании включена в уровень ASN1, в то время как традиционный формат включает ее на уровне PEM.

7.6.12.5 Примеры

Перевести закрытый ключ из традиционного формата в формат PKCS#5 v2.0 с помощью тройного алгоритма DES:

```
openssl pkcs8 -in key.pem -topk8 -v2 des3 -out enckey.pem
```

Перевести закрытый ключ в формат PKCS#8, используя алгоритм, совместимый с версией 1.5 формата PKCS#5:

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem
```

Перевести закрытый ключ в PKCS#8, используя алгоритм, совместимый с форматом PKCS#12 (3DES):

```
openssl pkcs8 -in key.pem -topk8 -out enckey.pem -v1 PBE-SHA1-3DES
```

Прочитать незашифрованный закрытый ключ формата PKCS#8 в DER-кодировке:

```
openssl pkcs8 -inform DER -nocrypt -in key.der -out key.pem
```

Перевести закрытый ключ из любого формата PKCS#8 в традиционный формат:

```
openssl pkcs8 -in pk8.pem -out key.pem
```

7.6.13 Команда `pkcs12`

7.6.13.1 Описание команды

Команда pkcs12 позволяет создавать и интерпретировать файлы формата PKCS#12 (иногда называемые файлами формата PFX). Файлы формата PKCS#12 используются для переноса между компьютерами закрытых ключей и соответствующих им сертификатов.

7.6.13.2 Формат ввода команды

openssl pkcs12 [-export] [-chain] [-inkey filename] [-certfile filename] [-name name] [-caname name] [-in filename] [-out filename] [-noout] [-nomacver] [-nocerts] [-clcerts] [-cacerts] [-nokeys] [-info] [-gost89 | -nodes] [-noiter] [-maciter | -nomaciter | -nomac] [-twopass] [-descert] [-certpbe cipher] [-keypbe cipher] [-macalg digest] [-keyex] [-keysig] [-password arg] [-passin arg] [-passout arg] [-rand file(s)] [-CAfile file] [-CApath dir] [-CSP name]

7.6.13.3 Опции команды

У данной команды имеется множество опций, значение некоторых зависит от того, создается или интерпретируется файл формата PKCS#12. По умолчанию считается, что файл интерпретируется. Можно создать файл формата PKCS#12, используя опцию -export (см. ниже).

7.6.13.3.1 Опции интерпретирования файлов

Опция	**Описание**
-in filename	Определяет имя интерпретируемого файла формата PKCS#12. По умолчанию файл считывается со стандартного ввода
-out filename	Файл для записи сертификатов и закрытых ключей, по умолчанию - стандартный вывод. Всё записывается в формате PEM

Опция	Описание
-password arg, -passin arg	Указывает, где содержится пароль для входного файла (т.е. файла формата PKCS#12). Для получения дополнительной информации по формату аргумента arg см. раздел 7.4
-passout arg	Источник пароля для зашифровывания любых полученных закрытых ключей. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4.
-noout	Отменяет вывод ключей и сертификатов (однако будет проверена корректность входного файла и возможность его распаковки)
-clcerts	Выводит только клиентские сертификаты (не выводит сертификаты УЦ)
-cacerts	Выводит только сертификаты УЦ (не выводит клиентские сертификаты)
-nocerts	Отменяет вывод любых сертификатов
-nokeys	Отменяет вывод закрытых ключей
-info	Выводит дополнительную информацию о структуре файла формата PKCS#12, использованных алгоритмах и количестве итераций
-gost89	Использовать алгоритм ГОСТ 28147-89 для зашифрования закрытых ключей перед выводом (при импорте ключей для российских криптографических алгоритмов следует использовать именно этот алгоритм шифрования)
-des	Использовать алгоритм DES для шифрования закрытых ключей перед выводом
-des3	Использовать алгоритм тройной DES для шифрования закрытых ключей перед выводом. Используется по умолчанию
-idea	Использовать алгоритм IDEA для шифрования закрытых ключей перед выводом
-aes128, -aes192, -aes256	Использовать алгоритм AES для шифрования закрытых ключей перед выводом
-camellia128, -camellia192, -camellia256	Использовать алгоритмы Camellia для шифрования закрытых ключе перед выводом
-nodes	Выводить закрытые ключи в незашифрованном виде
-nomacver	Не проверять целостность контейнера

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-twopass	Запрашивать у пользователя отдельные пароли для контроля целостности и для расшифрования

7.6.13.3.2 Опции создания файлов

Опция	Описание
-export	Указывает, что файл формата PKCS#12 будет создан, а не интерпретирован
-out filename	Указывает имя создаваемого файла формата PKCS#12. По умолчанию используется стандартный вывод
-in filename	Указывает имя входного файла, из которого следует считать сертификаты и закрытые ключи, по умолчанию используется стандартный вход. Они все должны быть в формате PEM. Порядок значения не имеет, но должен присутствовать один закрытый ключ и соответствующий ему сертификат. Если присутствуют дополнительные сертификаты, они также будут включены в файл формата PKCS#12
-inkey filename	Файл, из которого считывается закрытый ключ. Если эта опция не указана, закрытый ключ должен присутствовать во входном файле
-name friendlyname	Указывает «дружественное имя» для сертификата и закрытого ключа. Обычно это имя указывается в выпадающих списках приложений, импортирующих файл
-certfile filename	имя файла, из которого следует считать дополнительные сертификаты
-caname friendlyname	Указывает «дружественное имя» для других сертификатов. Эта опция может быть использована несколько раз, чтобы определить имена всех сертификатов в порядке их появления. Netscape игнорирует дружественные имена других сертификатов, а MSIE их показывает
-password arg, -passout arg	Указывает, где содержится пароль для выходного файла (т.е. файла формата PKCS#12). Для получения дополнительной информации по формату аргумента arg см. раздел 7.4
-passin password	Источник пароля для расшифровывания и ввода закрытых ключей. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4

Опция	Описание
-chain	Если эта опция указана, делается попытка включить всю цепочку сертификатов для пользовательского сертификата. Для поиска сертификатов используется стандартное хранилище сертификатов УЦ. Если найти все необходимые сертификаты не удается, это считается фатальной ошибкой
-keypbe alg, -certpbe alg	Эти опции позволяют выбрать алгоритм, используемый для зашифрования закрытого ключа и сертификатов. При экспорте ключей российских криптографических алгоритмов Следует указать gost89 в качестве параметра alg
-keyex|-keysig	Указывает, следует ли использовать закрытый ключ для обмена ключами или только для подписывания. Эта опция интерпретируется только MS Internet Explorer и подобными приложениями MS. Опция -keysig указывает, что ключ можно использовать только для подписи. Ключи, используемые только для подписи, можно использовать в подписывании сообщений формата S/textbackslash MIME, контрольной подписи Active X и в аутентификации SSL-клиента, хотя из-за ошибки в коде только Internet Explorer начиная с версии 5.0 поддерживает использование ключей только для подписи для аутентификации SSL-клиента
-macalg digest	Указывает алгоритм дайджеста, используемого для контроля целостности контейнера. При экспорте ключей российских криптографических алгоритмов следует указать md_gost12_512, md_gost12_256 или md_gost94 (последний только в режиме совместимости со старыми версиями, см. раздел 7.6.13.6
-nomaciter, -noiter	Устанавливают число итераций при вычислении кода аутентификации и ключа шифрования в значение 1. Эти опции следует использовать только в том случае, если вы хотите создать файлы, совместимые с Internet Explorer 4.0
-maciter	Устанавливает число итераций при вычислении кода аутентификации в умолчательное значение (2048). Опция включена для совместимости с предыдущими версиями OpenSSL.
-nomac	Создать контейнер без контроля целостности
-CAfile file	Файл, содержащий сертификаты доверенных удостоверяющих центров
-CApath dir	Каталог, содержащий сертификаты доверенных удостоверяющих центров. Этот каталог должен быть стандартным каталогом сертификатов, то есть с каждым сертификатом должна быть связана хэш-сумма поля subject name

Опция	Описание
-CSP name	Записывает значение параметра name в формате Microsoft CSP
-twopass	Запрашивает у пользователя отдельные пароли для контроля целостности и для зашифрования: большая часть приложений всегда предполагает, что они совпадают, так что эта опция лишает их возможности прочесть создаваемый файл формата PKCS#12

7.6.13.4 Примечания

Хотя у этой команды существует большое количество опций, большинство из них используется очень редко. Для интерпретации файла формата PKCS#12 достаточно использовать только опции -in и -out, для создания - также опции -export и -name.

Если не указана ни одна из опций -clcerts, -cacerts или -nocerts, все сертификаты будут выведены в порядке, в котором они находятся во входных файлах формата PKCS#12. Нет гарантии, что первый присутствующий сертификат соответствует закрытому ключу. Некоторые приложения, требующие закрытый ключ и сертификат, предполагают, что первый сертификат в файле соответствует закрытому ключу, поэтому другой порядок сертификатов может стать проблемой. Эту проблему решает опция -clcerts, выводящая только сертификат, соответствующий закрытому ключу. Если требуются сертификаты УЦ, их можно вывести в отдельный файл, используя опции -nokeys и -cacerts, чтобы вывести только сертификаты УЦ.

Алгоритмы -keypbe и -certpbe позволяют указать конкретные алгоритмы для зашифования закрытых ключей и сертификатов. В МагПро КриптоПакет в этом качестве используется алгоритм gost89.

7.6.13.5 Особенности работы с ключами российских криптографических алгоритмов

При экспорте ключей российских криптографических алгоритмов для шифрования и контроля целостности данных контейнера PKCS#12 следует использовать только алгоритмы ГОСТ. Поэтому в обязательном порядке должны быть указаны опции -keypbe и -macalg, про этом в опции -keypbe должно быть указано значение gost89, а в опции -macalg - значение md_gost12_256 или md_gost12_512. Если дополнительно указывается опция -certpbe, предписывающая шифровать не только закрытые ключи, но и сертификаты, в этой опции также следует указывать значение gost89.

При импорте ключей российских криптографических алгоритмов следует указывать либо опцию -gost89 (шифровать файл закрытого ключа алгоритмом ГОСТ 28147-89), либо опцию -nodes (не шифровать файл закрытого ключа). Использование других алгоритмов шифрования файла закрытого ключа не допускается.

7.6.13.6 Совместимость с предыдущими версиями «МагПро КриптоПакет»

СКЗИ «МагПро КриптоПакет» 3.0 реализует создание и проверку кода аутентификации контейнеров PKCS#12 с ключами российских криптографических алгоритмов в соответствии с Методическими рекомендациями технического комитета по стандартизации «Криптографическая защита информации» (ТК 26). Предыдущие версии СКЗИ «МагПро КриптоПакет» были

разработаны до принятия указанных рекомендаций, поэтому вырабатывают код аутентификации по другому алгоритму. Поэтому контейнер, созданный версией 3.0, не будет читаться предыдущими версиями «МагПро КриптоПакет» и наоборот.

Чтобы включить режим совместимости с предыдущими версиями «МагПро КриптоПакет», нужно установить переменную окружения LEGACY_GOST_PKCS12. При экспорте ключей российских криптографических алгоритмов в режиме совместимости с предыдущими версиями в опции -certpbe следует указывать значение md_gost94.

7.6.13.7 Примеры

Интерпретировать файл формата PKCS#12 и вывести его в файл:

```
openssl pkcs12 -in file.p12 -out file.pem -gost89
```

Вывести только клиентские сертификаты в файл:

```
openssl pkcs12 -in file.p12 -clcerts -out file.pem
```

Не зашифровывать закрытый ключ:

```
openssl pkcs12 -in file.p12 -out file.pem -nodes
```

Вывести информацию о файле формата PKCS#12:

```
openssl pkcs12 -in file.p12 -info -noout
```

Создать файл формата PKCS#12:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name <<My
Certificate>> -keypbe gost89 -macalg md_gost12_256
```

Включить дополнительные сертификаты:

```
openssl pkcs12 -export -in file.pem -out file.p12 -name <<My
Certificate>> -keypbe gost89 -macalg md_gost12_256 -certfile
othercerts.pem
```

7.6.14 Команда `pkey`

7.6.14.1 Описание команды

Команда предназначена для работы с закрытыми и открытыми ключами. Их можно преобразовывать в различные кодировки и выводить их компоненты.

7.6.14.2 Формат ввода команды

openssl pkey [-inform PEM|DER|ENGINE] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-cipher] [-text] [-text_pub] [-noout] [-pubin] [-pubout] [-engine cryptocom]

7.6.14.3 Опции команды

Опция	**Описание**
-inform DER|PEM|ENGINE	Указывает входную кодировку DER или PEM. Если указана -inform ENGINE, ключ будет прочитан с аппаратного ключевого носителя, указанного в параметре -in.

Опция	Описание
-outform DER|PEM	Указывает выходную кодировку DER или PEM.
-in filename	Указывает входной файл, из которого следует считать ключ. Если эта опция не указана, ключ считывается со стандартного ввода. Если ключ зашифрован, после введения команды последует запрос на пароль для расшифровки. Если -inform имеет значение ENGINE, в параметре -in должен быть передан идентификатор ключа на аппаратном ключевом носителе.
-passin arg	Источник пароля для входного файла. За дополнительной информацией о формате аргумента arg см. раздел 7.4.
-out filename	Указывает выходной файл, в который следует записывать ключ. Если эта опция не указана, ключ выводится в стандартный вывод. Если указаны какие-либо опции зашифрования, после введения команды последует запрос на пароль для расшифровки. Имя выходного файла не должно совпадать с именем входного файла.
-passout password	Источник пароля для выходного файла. За дополнительной информацией о формате аргумента arg см. раздел 7.4.
-cipher	Эти опции зашифровывают закрытый ключ с помощью указанного алгоритма. Может быть указано любое наименование алгоритма, которое принимает функция EVP_get_cipherbyname(), например des3.
-text	Выводит различные компоненты открытого или закрытого ключа открытым текстом в дополнение к закодированной версии.
-text_pub	Если обрабатываются и закрытый и открытый ключ, выводит компоненты только открытого ключа.
-noout	Запрещает вывод закодированной версии ключа.
-pubin	По умолчанию из входного файла считывается закрытый ключ. Если указана данная опция, из входного файла считывается открытый ключ.
-pubout	По умолчанию выводится закрытый ключ. Если указана данная опция, будет выведен открытый ключ. Эта опция устанавливается автоматически, если на вход подается открытый ключ.
-engine cryptocom	Этот параметр необходимо указывать, если ключ читается с аппаратного ключевого носителя.

7.6.14.4 Примеры

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Удалить пассфразу из закрытого ключа:

```
openssl pkey -in key.pem -out keyout.pem
```

Зашифровать закрытый ключ на пассфразе, используя алгоритм ГОСТ 28147-89:

```
openssl pkey -in key.pem -des3 -out keyout.pem
```

Преобразовать закрытый ключ из кодировки PEM в DER:

```
openssl pkey -in key.pem -outform DER -out keyout.der
```

Вывести компоненты закрытого ключа в стандартный вывод:

```
openssl pkey -in key.pem -text -noout
```

Вывести открытые компоненты закрытого ключа в стандартный вывод:

```
openssl pkey -in key.pem -text\_pub -noout
```

Просто вывести открытую часть закрытого ключа в файл:

```
openssl pkey -in key.pem -pubout -out pubkey.pem
```

7.6.15 Команда `pkeyparam`

7.6.15.1 Описание команды

Команда pkeyparam обрабатывает параметры ключевых пар. Их можно преобразовывать в различные кодировки и выводить их компоненты.

7.6.15.2 Формат ввода команды

openssl pkeyparam [-in filename] [-out filename] [-text] [-noout] [-engine id]

7.6.15.3 Опции команды

Опция	**Описание**
-in filename	Указывает имя вводного файла, из которого следует считывать параметры. Если эта опция не указана, параметры считываются со стандартного ввода.
-out filename	Указывает имя выводного файла, в который следует записыать параметры. Если эта опция не указана, параметры выводятся в стандартный вывод.
-text	Выводит параметры открытым текстом в дополнение к закодированной версии.
-noout	Запрещает вывод закодированной версии параметров.

Опция	Описание
-engine id	Указание модуля engine (по его уникальной идентификационной строке) заставляет команду pkeyparam попытаться получить функциональную ссылку на указанный модуль, при необходимости инициализируя его. Затем модуль будет установлен в качестве умолчательного для всех доступных алгоритмов.

7.6.15.4 Пример

Вывести текстовую версию параметров:

```
openssl pkeyparam -in param.pem -text
```

7.6.15.5 Примечания

У этой команды нет опций -inform или -outform, потому что поддерживается только кодировка PEM, так как тип ключа определяется PEM-заголовками.

7.6.16 Команда `pkeyutl`

7.6.16.1 Описание команды

Команду pkeyutl можно использовать для выполнения различных операций над открытым ключом при помощи указанного алгоритма. Команда предоставляет доступ к низкоуровневым API, использование которых в приложении требует экспертизы корректности встраивания, поэтому эту команду следует использовать только в отладочных или диагностических целях.

7.6.16.2 Формат ввода команды

openssl pkeyutl [-in file] [-out file] [-sigfile file] [-inkey file] [-keyform PEM|DER] [-passin arg] [-peerkey file] [-peerform PEM|DER] [-pubin] [-certin] [-rev] [-sign] [-verify] [-verifyrecover] [-encrypt] [-decrypt] [-derive] [-pkeyopt opt:value] [-hexdump] [-asn1parse] [-engine id]

7.6.16.3 Опции команды

Опция	Описание
-in filename	Указывает имя вводного файла, из которого следует считывать данные. Если эта опция не указана, данные считываются со стандартного ввода.
-out filename	Указывает имя выводного файла, в который следует записыать данные. По умолчанию данные выводятся в стандартный вывод.
-inkey file	Входной ключевой файл, по умолчанию это должен быть файл закрытого ключа.

Опция	Описание
-keyform PEM|DER	Формат ключа PEM, DER или ENGINE.
-passin arg	Источник пароля для входного файла. За дополнительной информацией о формате аргумента arg см. раздел 7.4.
-peerkey file	Файл ключа партнера по взаимодействию, используется в операциях согласования ключей и выработки общего секрета.
-peerform PEM|DER	Формат ключа партнера по взаимодействию PEM, DER или ENGINE.
-engine id	Указание модуля engine (по его уникальной идентификационной строке) заставляет команду pkeyutl попытаться получить функциональную ссылку на указанный модуль, при необходимости инициализируя его. Затем модуль будет установлен в качестве умолчательного для всех доступных алгоритмов.
-pubin	Входной файл — открытый ключ.
-certin	Входной файл — сертификат, содержащий открытый ключ.
-rev	Обращает порядок входного буфера. Это полезно для некоторых библиотек (таких, как CryptoAPI), которые представляют буфер в формате little endian.
-sign	Подписывает входные данные и выводит подписанный результат. Для выполнения данной операции требуется закрытый ключ.
-verify	Проверяет подпись под указанными данными и выводит результаты проверки.
-verifyrecover	Проверяет входные данные и выводит исходные данные.
-encrypt	Зашифровывает входные данные с использованием открытого ключа.
-decrypt	Расшифровывает входные данные с использованием закрытого ключа.
-derive	Извлекает общий секрет с использованием ключа партнера по взаимодействию.
-hexdump	Выводит шестнадцатеричный дамп выходных данных.
-asn1parse	Разбирает ASN.1-структуру выходных данных. Это полезно в сочетании с опцией -verifyrecover, когда подписывается ASN.1-структура.

7.6.16.4 Примечания

Для различных алгоритмов и их приложений поддерживаются различные наборы операций и опций. Алгоритм ГОСТ Р 34.10 поддерживает операции encrypt, decrypt, sign, verify, verifyrecover и derive.

7.6.16.5 Примеры

Подписать данные с использованием закрытого ключа:

```
openssl pkeyutl -sign -in file -inkey key.pem -out sig
```

Вывести подписанные данные (с использованием ключа ГОСТ)

```
openssl pkeyutl -verifyrecover -in sig -inkey key.pem
```

Проверить подпись

```
openssl pkeyutl -verify -in file -sigfile sig -inkey key.pem
```

Сформировать значение общего секрета:

```
openssl pkeyutl -derive -inkey key.pem -peerkey pubkey.pem -out secret
```

7.6.17 Команда `rand`

7.6.17.1 Описание команды

Команда rand выводит определенное количество (указанное в параметре num) псевдослучайных байт после однократной инициализации генератора случайных чисел. Как и в других командах утилиты openssl, инициализация ДСЧ использует файл $HOME.rnd или .rnd в дополнение к файлам, указанным в опции -rand. Новый файл $HOME.rnd или .rnd будет записан, если из этих источников получено достаточно случайного материала для инициализации.

7.6.17.2 Формат ввода команды

openssl rand [-out file] [-rand file(s)] [-base64] [-hex] num

7.6.17.3 Опции команды

Опция	**Описание**
-out file	Записывает в файл с именем file, а не в стандартный вывод
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-base64	Выполняет кодирование выходных данных в формате base64

Опция	Описание
-hex	Представляет выходные данные в виде шестнадцатиричной строки

7.6.18 Команда `req`

7.6.18.1 Описание команды

Команда req в основном используется для создания и обработки заявок на сертификаты формата PKCS#10. Она также может создавать самоподписанные сертификаты, которые можно использовать, например, в качестве корневых сертификатов удостоверяющих центров.

Внимание. При использовании «МагПро КриптоПакет» 3.0 команду req утилиты openssl можно использовать для создания ключей. Но следует иметь в виду, что эта команда записывает ключи только в PKCS#8-контейнеры. Для создания ключей, которые записываются в аппаратные устройства («Аккорд», «Соболь»), следует использовать программу mkkey из состава «МагПро КриптоПакет» 3.0. Заявки на регистрацию ключей, созданных с помощью программы mkkey, создаются с помощью команды req утилиты openssl. Кроме того, создание ключей с помощью команды req возможно только при наличии установленного на компьютере аппаратного ДСЧ: использование клавиатурного датчика в этом случае невозможно. Программа mkkey позволяет создавать ключи при помощи клавиатурного датчика.

7.6.18.2 Формат ввода команды

openssl req [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-text] [-pubkey] [-noout] [-verify] [-modulus] [-new] [-rand file(s)] [-newkey alg:file] [-nodes] [-key filename] [-keyform PEM|DER] [-keyout filename] [-keygen_engine id] [-[digest]] [-config filename] [-subject] [-subj arg] [-multivalue-rdn] [-x509] [-days n] [-set_serial n] [-asn1-kludge] [-newhdr] [-extensions section] [-reqexts section] [-utf8] [-nameopt] [-reqopt] [-batch] [-verbose] [-engine id]

7.6.18.3 Опции команды

Опция	Описание
-inform DER|PEM	Определяет входной формат. Опция DER использует ASN.1 DER-закодированную форму, совместимую с PKCS#10. Форма PEM — формат по умолчанию: она состоит из DER-формы, закодированной в base64, с дополнительными верхним и нижним ограничителями.
-outform DER|PEM	Определяет выходной формат, опция имеет те же значения, что и опция -inform.
-in filename	Определяет входной файл, из которого следует считывать заявку. Если эта опция не указана, заявка считывается со стандартного входа. Заявка считывается только в том случае, если не указаны опции создания (-new и -newkey).
-passin arg	Источник пароля для входного файла. За дополнительной информацией о формате аргумента arg см. раздел 7.4.

Опция	Описание
-out filename	Указывает имя выходного файла для записи результатов выполнения команды. По умолчанию используется стандартный вывод.
-passout arg	Источник пароля для выходного файла. За дополнительной информацией о формате аргумента arg см. раздел 7.4.
-text	Выводит заявку в текстовом виде.
-subject	Выводит значение поля subject заявки (или сертификата, если указана опция -x509)
-pubkey	Выводит открытый ключ.
-noout	Эта опция предотвращает вывод закодированной версии заявки.
-modulus	Эта опция выводит значение модулюса открытого ключа, содержащегося в заявке.
-verify	Проверяет подпись под заявкой.
-new	Эта опция создает новую заявку на сертификат. Она запрашивает у пользователя значения соответствующих полей. Конкретные запрашиваемые поля, а также их максимальные и минимальные размеры определяются в конфигурационном файле, как и любые запрашиваемые расширения. Если опция -key не указана, данная опция создаст новый закрытый ключ RSA, используя информацию, содержащуюся в конфигурационном файле.
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-newkey alg	Эта опция создает новую заявку на сертификат и новый закрытый ключ. Аргумент alg может иметь несколько форм. Для алгоритма ГОСТ Р 34.10 возможны следующие формы: *алгоритм:файл* Использование аргумента алгоритм:filename создает ключ алгоритма ГОСТ Р 34.10 (требует конфигурирования модуля engine cryptocom в конфигурационном файле). *алгоритм* (параметры указываются в опции -pkeyopt) Если указать только алгоритм gost2012_256, gost2012_512 или gost2001, следует указать набор параметров в опции -pkeyopt, например -pkeyopt paramset:X. Поддерживаются следующие наборы параметров: – для gost2012_512: A, B; – для gost2012_256 и gost2001: – для ключей подписи: A, B, C; – для ключей обмена ключами: XA, XB.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-pkeyopt opt:value	Устанавливает опцию алгоритма открытого ключа в значение value. Точный набор поддерживаемых опций зависит от используемого алгоритма открытого ключа и его реализации. Подробности см. в разделе 7.6.9.4.
-key filename	Указывает файл, из которого следует считать закрытый ключ.
-keyform PEM|DER	Формат закрытого ключа, указанного в качестве аргумента опции -key. По умолчанию PEM.
-keyout filename	Указывает файл, в который следует записать созданный закрытый ключ. Если эта опция не указана, используется файл, указанный в конфигурационном файле.
-nodes	Если эта опция указана, то если создается закрытый ключ, он не зашифровывается.
-[digest]	Указывает алгоритм хэширования, с помощью которого следует подписать заявку. В случае использования алгоритма подписи ГОСТ Р 34.10 в качестве алгоритма дайджеста всегда используется GOST R 34.11, что бы ни было указано в данной опции, поэтому при использовании алгоритмов ГОСТ данную опцию можно не указывать.
-config filename	Это позволяет указать альтернативный конфигурационный файл. Данная опция имеет больший приоритет, чем имя файла, заданное при компиляции или имя, указанное в переменной среды OPENSSL_CONF.
-subj arg	Устанавливает значение поля subject для новой заявки или замещает значение этого поля при обработке заявки. Формат аргумента arg должен быть /type0=value0/type1=value1/type2=..., символы могут быть экранированы знаком \(обратный слэш), пробелы не опускаются.
-multivalue-rdn	Эта опция указывает, что аргумент опции -subj необходимо интерпретировать с полной поддержкой многозначных RDN. Пример: /DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe Если опция -multi-rdn не используется, то значение поля UID будет 123456+CN=John Doe.
-x509	Эта опция создает самоподписанный сертификат вместо заявки. Это обычно используется для создания тестового сертификата или самоподписанного корневого сертификата удостоверяющего центра. Расширения, добавляемые в сертификат (если таковые есть) указываются в конфигурационном файле. Если не указано другого с помощью опции set_serial, в качестве серийного номера будет указан 0.
-days n	Если указана опция -x509, данная опция указывает срок действия сертификата в днях. По умолчанию 30 дней.
-set serial n	Серийный номер для выпускаемого самоподписанного сертификата. Может быть указан как десятичная величина, или как шестнадцатиричная с префиксом 0x. Указывать отрицательные серийные номера можно, но не рекомендуется.

Опция	Описание
-extensions section -reqexts section	Эти опции указывают альтернативные секции для включения расширений в сертификат (если указана опция -x509) или в заявку. Это позволяет использовать несколько различных секций в одном и том же конфигурационном файле для создания заявок с различными целевыми назначениями.
-utf8	Эта опция указывает, что значения полей следует интерпретировать как строки в кодировке UTF8, по умолчанию они интерпретируются в кодировке ASCII. Это означает, что значения полей, считанные с терминала или из конфигурационного файла, должны быть корректными UTF-8 строками.
-nameopt option	Опция указывает, как должны выводиться значения полей subject и issuer. Аргументом может быть одна опция или несколько, разделенных запятыми. Для установки нескольких опций можно также несколько раз использовать свитч -nameopt. Для дополнительной информации см. раздел 7.6.28.
-reqopt option	Опция управляет форматом вывода, используемым с опцией -text. Аргументом может быть одна опция или несколько, разделенных запятыми. Для дополнительной информации см. описание опции -certopt в разделе 7.6.28.
-asn1-kludge	По умолчанию команда req выводит заявки, не содержащие атрибутов, в корректном формате PKCS#10. Но некоторые удостоверяющие центры принимают только заявки, не содержащие атрибутов, в некорректном формате. Эта опция создает такой некорректный формат. Точнее атрибуты в заявке формата PKCS#10 определены как атрибут SET OF. Они не являются опциональными, поэтому, если атрибуты в заявке отсутствуют, они должны быть закодированы как пустой SET OF. Эта некорректная форма не включает такой пустой SET OF, а корректная форма включает. Следует отметить, что очень немногие удостоверяющие центры требуют использования данной опции.
-no-asn1-kludge	Опция, противоположная -asn1-kludge.
-newhdr	Добавляет слово NEW в верхний и нижний ограничители в PEM-файле заявки. Это необходимо для некоторых программ (сертификационный сервер Netscape) и некоторых удостоверяющих центров.
-batch	Неинтерактивный режим.
-verbose	Вывести дополнительную информацию о производимых операциях.
-engine id	выбор энджина (по его уникальной строке id) приводит к тому, что req делает попытку получить функциональную ссылку на выбранный энджин, таким образом инициализируя его, если это необходимо. После этого энджин устанавливается, как энджин по умолчанию, для всех доступных алгоритмов.
-keygen_engine id	Указывает загружаемый модуль engine (по его уникальной идентификационной строке), который будет использоваться для операций создания ключей.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.18.4 Формат конфигурационного файла

Опции конфигурации указываются в разделе req конфигурационного файла. Как и в любом конфигурационном файле, если некая величина не указана в конкретном разделе (например, разделе req) то она ищется также в начальном непоименованном разделе или в разделе по умолчанию.

Доступные опции подробно описаны ниже.

Опция	**Описание**
input_password output_password	Пароли для входного файла закрытого ключа (если таковой присутствует) и для выходного файла закрытого ключа (если таковой создается). Опции командной строки passin и passout имеют больший приоритет, чем опции, указанные в конфигурационном файле.
default_bits	Эта опция определяет умолчательный размер ключа в битах. Если опция не указана, используется 512. Опция применяется, если в командной строке указана опция -new. Опция командной строки -newkey имеет больший приоритет.
default_keyfile	Это умолчательное имя для выходного файла закрытого ключа. Если эта опция не указана, ключ записывается в стандартный выход. Опция командной строки -keyout имеет больший приоритет.
oid_file	Эта опция указывает на файл, содержащий дополнительные OID (OBJECT IDENTIFIERS). Каждая строка файла должна иметь следующий формат: OID в численном виде, пробел, короткое имя, пробел, длинное имя.
oid_section	Эта опция указывает на раздел конфигурационного файла, содержащий дополнительные OID. Каждая строка раздела должна иметь формат: короткое имя OID=численный вид OID. В случае использования этой опции короткое и длинное имена совпадают.
RANDFILE	Файл, используемый для считывания и записи информации для инициализации генератора случайных чисел.
encrypt_key	Если эта опция установлена в 0, то создаваемый закрытый ключ не зашифровывается. Эта опция эквивалентна опции командной строки -nodes.
default_md	Опция обязательно должна быть указана, но используется только в том случае, если ключ удостоверяющего центра имеет алгоритм, позволяющий использовать разные алгоритмы хэширования (RSA). Значение опции представляет собой название алгоритма хэширования, используемого для подписи сертификатов (в противном случае оно может быть любым).

Опция	Описание
string_mask	Эта опция маскирует использование некоторых типов строк в некоторых полях. Существует несколько возможных значений данной опции. Значение default (оно же по умолчанию) использует PrintableStrings, T61Strings и BMPStrings. При указании значения pkix будут использоваться только PrintableStrings and BMPStrings в соответствии с PKIX рекомендацией в RFC2459. Если указывается значение utf8only, используются только UTF8Strings: это рекомандация PKIX в RFC2459 после 2003 г. Наконец, значение nombstr использует только PrintableStrings и T61Strings: некоторые программы встречают затруднения с BMPStrings and UTF8Strings, особенно Netscape. Для поддержки кириллицы в полях сертификата необходимо указывать либо значение pkix (тогда формируемые заявки будут использовать тот же набор строковых типов, что и заявки, создаваемые Active-X элементом Xenroll в Windows), либо utf8only
req_extensions	Указывает раздел конфигурационного файла, содержащий список расширений, которые необходимо добавить в заявку на сертификат. Командно-строчный свитч -reqexts имеет больший приоритет.
x509_extensions	Указывает раздел конфигурационного файла, содержащий список расширений, которые необходимо добавить в сертификат, созданный с использованием опции -x509. Командно-строчный свитч -extensions имеет больший приоритет.
prompt	Если эта опция имеет значение no, отключается запрашивание полей сертификата и просто считывает значения полей прямо с конфигурационного файла. Кроме того, изменяется ожидаемый формат разделов distinguished_name и attributes.
utf8	Эта опция указывает, что значения полей следует интерпретировать как строки в кодировке UTF8, по умолчанию они интерпретируются в кодировке ASCII. Это означает, что значения полей, считанные с терминала или из конфигурационного файла, должны быть корректными UTF-8 строками.
attributes	Указывает раздел конфигурационного файла, содержащий атрибуты заявки: его формат совпадает с форматом distinguished_name. Как правило, они содержат типы challengePassword или unstructuredName. В настоящее время они игнорируются утилитами OpenSSL, выполняющими подписывание заявки, но некоторые удостоверяющие центры могут требовать их наличия.
distinguished_name	Указывает раздел конфигурационного файла, содержащий поля структуры distinguished name, которые следует запрашивать при создании сертификата или заявки. Формат описан в разделе 7.6.18.5.

7.6.18.5 Формат разделов конфигурационного файла distinguished name и attribute

Существуют два различных формата для разделов distinguished name и attribute. Если опция prompt установлена в значение no, эти разделы просто содержат наименования и значения полей, например

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

CN=Ivanov Ivan Ivanovich OU=Company emailAddress=someone@somewhere.org

Это позволяет внешним программам (например, программам с графическим интерфейсом) создавать файл-шаблон со всеми названиями и значениями полей и просто передавать этот файл команде req. Пример такого рода конфигурационного файла содержится в разделе .

Или же, если опция prompt не указана или не установлена в no, файл содержит информацию о запросах полей. Она состоит из строк вида:

```
fieldName="prompt"
fieldName_default="значение поля по умолчанию"
fieldName_min= 2
fieldName_max= 4
```

Здесь fieldName — наименование используемого поля, например commonName или CN. Строка символов `"prompt"` используется для запроса к пользователю ввести соответствующую информацию. Если пользователь ничего не вводит, используется умолчательное значение поля. Если и умолчательного значения не указано, поле опускается. Поле может быть опущено и в том случае, если величина по умолчанию присутствует, но пользователь просто введет символ '.'.

Количество введенных символов должно быть в пределах fieldName_min and fieldName_max: могут также быть дополнительные ограничения в зависимости от рассматриваемого поля (например, значение поля countryName может быть только двухбуквенным и соответствовать типу PrintableString).

Некоторые поля (например organizationName) могут использоваться в структуре DN больше одного раза. Это представляет собой проблему, потому что конфигурационные файлы не распознают одно и то же имя, встречающееся дважды. Чтобы избежать этой проблемы, если fieldName содержит несколько символов, за которыми следует точка, они будут проигнорированы. Поэтому, например, второе поле organizationName может быть введено как 1.organizationName.

Корректные разрешенные наименования полей могут быть любыми короткими или длинными именами OID. Они компилируются в OpenSSL и включают обычные величины, такие как commonName, countryName, localityName, organizationName, organizationUnitName, stateOrProvinceName. Дополнительно введены также emailAddress, name, surname, givenName initials и dnQualifier.

Дополнительные OID могут быть определены с помощью опций конфигурационного файла oid_file и oid_section. Любые дополнительные поля обрабатываются как строки типа DirectoryString.

7.6.18.6 Примеры

Просмотреть и проверить заявку на сертификат:

```
openssl req -in req.pem -text -verify -noout
```

Генерировать заявку на сертификат с явным указанием ключа:

```
openssl req -new -key key.pem -out req.pem
```

То же самое, но с генерацией ключа:

```
openssl req -newkey gost2012_256:A -keyout key.pem -out req.pem
```

Создать самоподписанный корневой сертификат:

```
openssl req -x509 -newkey gost2012_256:A -keyout key.pem -out
req.pem
```

Пример файла, который указывается в опции oid_file:

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
1.2.3.4         shortName       A longer Name
1.2.3.6         otherName       Other longer Name
```

Пример раздела конфигурационного файла, который указывается в опции oid_section с использованием переменного расширения:

```
testoid1=1.2.3.5
testoid2=${testoid1}.6
```

Образец конфигурационного файла, обеспечивающего вывод запросов значений полей:

```
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions         = v3_ca

dirstring_type = nobmp

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2

localityName                    = Locality Name (eg, city)

organizationalUnitName          = Organizational Unit Name (eg, section)

commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 40

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20

[ v3_ca ]

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```

Образец конфигурационного файла с указанными значениями полей:

```
RANDFILE                = $ENV::HOME/.rnd

[ req ]
default_bits            = 1024
default_keyfile         = keyfile.pem
distinguished_name      = req_distinguished_name
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
attributes              = req_attributes
prompt                  = no
output_password         = mypass

[ req_distinguished_name ]
C                       = GB
ST                      = Test State or Province
L                       = Test Locality
O                       = Organization Name
OU                      = Organizational Unit Name
CN                      = Common Name
emailAddress            = test@email.address

[ req_attributes ]
challengePassword               = A challenge password
```

7.6.18.7 Примечания

Как правило, верхний и нижний ограничители в формате PEM выглядят как:

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

Некоторым программам (например некоторым версиям сертификационного сервера Netscape) необходим другой вид ограничителей:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
-----END NEW CERTIFICATE REQUEST-----
```

Такие ограничители создаются при использовании опции -newhdr, но в обратную сторону они совместимы. Обе формы при вводе принимаются прозрачно.

Заявки на сертификаты, создаваемые в Microsoft IE Active-X элементом Xenroll, включают в себя добавленные расширения, в том числе расширение KeyUsage, которое определяет тип ключа (только подпись или общего назначения) и все дополнительные OID, введенные скриптом в расширении extendedKeyUsage.

7.6.18.8 Диагностика

Часто выводятся следующие сообщения:
Using configuration from /some/path/openssl.cnf
Unable to load config info
Через некоторое время выводится:
unable to find 'distinguished_name' in config
problems making Certificate Request
Первое сообщение — ключевое: не обнаружен конфигурационный файл! Некоторые операции (такие как просмотр заявки на сертификат) не требуют конфигурационного файла, поэтому его использование необязательно. Но создание сертификатов или заявок требует конфигурационного файла. Это можно считать ошибкой.
Еще одно озадачивающее сообщение:
Attributes:
a0:00

Это сообщение выводится, когда никаких attributes не указано, а заявка включает корректную пустую структуру SET OF (DER-форма которой выглядит как 0xa0 0x00). Если вы видите только:

Attributes:

Значит, структура SET OF отсутствует и кодировка технически некорректна (но допустима). Для получения дополнительной информации см. описание опции -asn1-kludge.

7.6.18.9 Переменные среды

Переменная `OPENSSL_CONF`, если она определена, позволяет определить расположение дополнительного конфигурационного файла. Опция командной строки -config имеет больший приоритет. Для совместимости переменная среды `SSLEAY_CONF` может использоваться для той же цели, но ее использование не рекомендуется.

7.6.19 Команда `s_client`

7.6.19.1 Описание команды

Команда s_client реализует безовый клиент SSL/TLS, который подключается к удаленному компьютеру с использованием SSL/TLS. Это очень полезный диагностический инструмент для SSL-серверов.

7.6.19.2 Формат ввода команды

openssl s_client [-connect host:port] [-servername name] [-verify depth] [-verify_return_error] [-cert filename] [-certform DER|PEM] [-key filename] [-keyform DER|PEM] [-pass arg] [-CApath directory] [-CAfile filename] [-no_alt_chains] [-reconnect] [-pause] [-showcerts] [-debug] [-msg] [-nbio_test] [-state] [-nbio] [-crlf] [-ign_eof] [-no_ign_eof] [-quiet] [-ssl2] [-ssl3] [-tls1] [-no_ssl2] [-no_ssl3] [-no_tls1] [-no_tls1_1] [-no_tls1_2] [-fallback_scsv] [-bugs] [-cipher cipherlist] [-serverpref] [-starttls protocol] [-engine id] [-tlsextdebug] [-no_ticket] [-sess_out filename] [-sess_in filename] [-rand file(s)] [-serverinfo types] [-status] [-nextprotoneg protocols]

7.6.19.3 Опции команды

Опция	**Описание**
-connect host:port	Указывает адрес сервера, с которым нужно установить соединение, и опционально порт. Если не указано другое, делается попытка установить связь с локальным хостом, порт 4433.
-servername name	Устанавливает расширение TLS SNI (Server Name Indication) в сообщении ClientHello.
-cert certname	Указывает сертификат, который следует использовать, если таковой запрашивается сервером. По умолчанию сертификат не используется.
-certform format	Формат используемого сертификата: DER или PEM. По умолчанию PEM.

Опция	Описание
-key keyfile	Закрытый ключ, который следует использовать. Если не указан, будет использоваться файл сертификата.
-keyform format	Формат закрытого ключа: DER или PEM. По умолчанию PEM.
-pass arg	Источник пароля для закрытого ключа. Для получения дополнительной информации о формате аргумента arg см. раздел 7.4.
-verify depth	Используемая глубина верификации. Опция определяет максимальную длину цепочки сертификатов и включает проверку серверного сертификата. В настоящее время операция проверки продолжается и после появления сообщений об ошибках, чтобы диагностировать все проблемы в цепочке сертификатов. В качестве побочного эффекта соединение не обрывается в случае, если серверный сертификат будет признан некорректным.
-verify_return_error	Возвращает ошибку верификации вместо продолжения работы. Обычно это приводит к прекращению соединения с фатальной ошибкой.
-CApath directory	Каталог, который следует использовать для проверки серверного сертификата. Этот каталог должен быть в «хэш-формате», см. раздел 7.6.26 для получения дополнительной информации. Сертификаты из этого каталога также используются для построения цепочки для проверки клиентского сертификата.
-CAfile file	Файл, содержащий доверенные сертификаты, которые следует использовать при аутентификации сервера и во время попыток построить цепочку клиентских сертификатов.
-purpose, -ignore_critical, -issuer_checks, -crl_check, -crl_check_all, -policy_check, -extended_crl, -x509_strict, -policy, -check_ss_sig, -no_alt_chains	Устанавливает различные опции проверки цепочки сертификатов. Для более детальной информации смотри опцию verify из руководства пользователя.
-reconnect	Указывает, что необходимо связываться с одним и тем же сервером 5 раз с одним и тем же сессионным ID. Эту опцию можно использовать при тестировании кэширования сессий.
-pause	Устанавливает односекундную паузу между каждым вызовом функций read и write.
-showcerts	Выводит всю цепочку сертификатов, присланную сервером. Как правило, выводится только сам сертификат сервера.

Опция	Описание
-prexit	Выводит информацию о сессии при завершении работы программы. При использовании этой опции попытка выведения информации о сессии предпринимается всегда, даже если установить соединение не удается. Эта опция полезна, потому что используемый шифр-сьют может быть пересогласован или соединение может быть не установлено из-за того, что требуется сертификат клиента или таковой запрашивается после попытки обратиться к определенному URL. Примечание: результат работы этой опции не всегда является точным, потому что соединение, возможно, так и не удастся установить.
-state	Выводит состояния SSL-сессии.
-debug	Выводит подробную отладочную информацию, включающую шестнадцатеричный дамп всего траффика.
-msg	Выводит все сообщения протокола с шестнадцатеричным дампом.
-nbio_test	Тестирует неблокирующий ввод-вывод
-nbio	Включает неблокирующий ввод-вывод
-crlf	Эта опция переводит символ конца строки с терминала в CR+LF, как требуют некоторые серверы.
-ign_eof	Подавляет закрытие соединения при обнаружении конца файла стандартного ввода. Включается по умолчанию, если указана опция -quiet.
-quiet	Подавляет вывод сессионной информации и информации о сертификате. Как следствие, отключает действие опции — -ign_eof.
-no_ign_eof	Закрывает соединение при обнаружении конца файла стандартного ввода, даже если указана опция -quiet.
-psk_identity identity	Использует PSK (Pre-Shared Key) идентификатор identity, когда используется шифр-сьют PSK.
-psk key	Использует ключ PSK key, когда используется шифр-сьют PSK. Ключ указывается, как шестнадцатеричное число без префикса 0x, например, -psk 1a2b3c4d.
-ssl2, -ssl3, -tls1, -tls1_1, -tls1_2, -no_ssl2, -no_ssl3, -no_tls1, -no_tls1_1, -no_tls1_2	Эти опции требуют или запрещают использование указанных протоколов SSL или TLS. По умолчанию при установлении соединения используется гибкий (version-flexible) метод, который позволяет выбрать максимальную взаимно поддерживаемую версию.
-fallback_scsv	Посылает TLS_FALLBACK_SCSV в сообщении ClientHello.
-bugs	В распространенных реализациях SSL и TLS есть несколько известных ошибок. Указание этой опции включает разнообразные методы их обхода.
-cipher cipherlist	Эта опция позволяет модифицировать список допустимых шифр-сьютов, отправляемый клиентом. Хотя сервер определяет, какой шифр-сьют следует использовать, он обязан использовать первый поддерживаемый шифр-сьют из отправленного клиентом списка. Для получения дополнительной информации см. руководство по команде ciphers.
-serverpref	Использует предпочтения шифров, выбранные сервером; используется только для SSLV2.

Опция	Описание
-starttls protocol	Отправляет протокольно-специфичное сообщение (сообщения) для переключения в TLS для коммуникации. Аргумент protocol - ключевое слово для соответствующего протокола. В настоящее время поддерживаются только ключевые слова smtp, pop3, imap, ftp и xmpp.
-tlsextdebug	Печатает расширения TLS, полученные от сервера, в шестнадцатеричном виде.
-no_ticket	Отключает поддержку сессионных тикетов по RFC4507bis.
-sess_out filename	Записывает SSL-сессию в файл filename output SSL session to filename
-sess_in sess.pem	Загружает SSL-сессию из файла. Клиент попытается возобновить соединение, которое было во время этой сессии.
-engine id	Указывает загружаемый модуль engine (по его уникальной идентификационной строке) и вызывает попытку использовать реализацию криптографических алгоритмов из этого модуля.
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-serverinfo types	Список типов расширений TLS (числа между 0 и 65535), разделенных символом запятой. Каждый тип расширения будет отправлен, как пустое расширение TLS в сообщении ClientHello. Ответ сервера (если будет) будет закодирован и отображен, как файл PEM.
-status	Запрашивает статус сертификата с сервера
-nextprotoneg protocols	Запускает TLS расширения Next Protocol Negotiation и позволяет указать список имен протоколов, разделенных запятой, которые клиент должен объявить, как поддерживающиеся. В начале этого списка должны быть наиболее востребованные протоколы. Имена протоколов указываются в кодировке ASCII, например, "http/1.1"или "spdy/3". Пустой список протоколов обрабатывается специальным образом и приводит к тому, что клиент заявляет поддержку расширения TLS, но сразу после получения сообщения ServerHello вместе со списком протоколов, поддерживаемых сервером, произойдет разрыв соединения.

7.6.19.4 Команды, выводимые при установленном соединении

Если установлено соединение с SSL-сервером, то выводятся все данные, полученные с сервера, и все нажатия на клавиши будут переданы на сервер. При интерактивном использовании (что означает, что не были указаны ни опция -quiet, ни опция -ign_eof) то сессия будет пересогласована, если строка начинается с R, а если строка начинается с Q, или достигнут конец файла, соединение будет закрыто.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.19.5 Примечания

Команда s_client может применяться для отладки SSL-сервером. Для установления соединения с SSL HTTP-сервером обычно используется команда:

`openssl s_client -connect servername:443` (протокол https использует порт 443). Если соединение установлено успешно, можно дать http-команду, например "GET /" — запрос веб-страницы.

Если «рукопожатие» неудачно, этому может быть несколько возможных причин, если ничего очевидного, вроде отсутствия клиентского сертификата, можно попробовать использовать опции -bugs, -ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1 в том случае, если ошибка на сервере. Особенно вы можете поиграть с этими опциями перед отправкой баг-репорта в список рассылки OpenSSL.

Часто встречающая проблема при попытке заставить работать клиентские сертификаты — веб-клиент жалуется, что у него нет сертификатов, или выдает пустой список для выбора. Это, как правило, происходит потому, что сервер не отправляет название удостоверяющего центра клиента в своем списке «приемлемых удостоверяющих центров», когда он запрашивает сертификат. При использовании программы s_client можно просмотреть и проверить список приемлемых удостоверяющих центров. Но некоторые серверы запрашивают клиентскую аутентификацию только после запроса определенного URL. Чтобы получить список в этом случае, необходимо воспользоваться опцией -prexit и отправить http-запрос соответствующей страницы.

Если сертификат указан в командной строке с использованием опции -cert, он не будет использоваться, если только сервер специально не запросит клиентский сертификат. Таким образом, простое включение клиентского сертификата в командную строку не является гарантией, что сертификат работает.

Если возникают трудности при просмотре серверного сертификата, следует воспользоваться опцией -showcerts, чтобы просмотреть всю цепочку сертификатов.

7.6.20 Команда `s_server`

7.6.20.1 Описание команды

Команда s_server реализует SSL/TLS-сервер общего назначения, который отвечает на запросы на установление соединения на определенном порте с использованием протокола SSL/TLS.

7.6.20.2 Формат ввода команды

openssl s_server [-accept port] [-context id] [-verify depth] [-Verify depth] [-crl_check] [-crl_check_all] [-cert filename] [-certform DER|PEM] [-key keyfile] [-keyform DER|PEM] [-pass arg] [-dcert filename] [-dcertform DER|PEM] [-dkey keyfile] [-dkeyform DER|PEM] [-dpass arg] [-dhparam filename] [-nbio] [-nbio_test] [-crlf] [-debug] [-msg] [-state] [-CApath directory] [-CAfile filename] [-no_alt_chains] [-nocert] [-cipher cipherlist] [-serverpref] [-quiet] [-no_tmp_rsa] [-ssl2] [-ssl3] [-tls1] [-no_ssl2] [-no_ssl3] [-no_tls1] [-no_dhe] [-bugs] [-hack] [-www] [-WWW] [-HTTP] [-engine id] [-tlsextdebug] [-no_ticket] [-id_prefix arg] [-rand file(s)] [-serverinfo file] [-no_resumption_on_reneg] [-status] [-status_verbose] [-status_timeout nsec] [-status_url url] [-nextprotoneg protocols]

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.20.3 Опции команды

Опция	Описание
-accept port	TCP-порт, который следует прослушивать в ожидании запросов на соединения. Если не указан, используется порт 4433.
-context id	Устанавливает идентификатор контекста SSL. В качестве значения может быть любая строка. Если эта опция не указана, используется значение по умолчанию.
-cert certname	Сертификат, который следует использовать. Большинство серверных шифр-сьютов требуют использования сертификатов, но некоторые требуют сертификат с определенным видом открытого ключа. Если не указано, используется имя файла server.pem.
-certform format	Формат используемого сертификата: DER или PEM. По умолчанию PEM.
-key keyfile	Закрытый ключ, который следует использовать. Если эта опция не указана, используется файл сертификата.
-keyform format	Формат используемого закрытого ключа: DER или PEM. По умолчанию PEM.
-pass arg	Источник пароля закрытого ключа. Для получения дополнительной информации о формате аргумента arg см. раздел 7.4.
-dcert filename, -dkey keyname	Указывает дополнительный сертификат и закрытый ключ, которые ведут себя так же, как сертификат и закрытый ключ, указанные в опциях -cert и -key, за исключением того, что если эти опции не указаны, никаких умолчательных дополнительных сертификата и ключа не используется. Как указано выше, некоторые шифр-сьюты требуют сертификат, содержащий ключ определенного типа. Поэтому если сервер уже работает с сертификатом другого типа, ему необходим дополнительный сертификат для установления соединения.
-dcertform format, -dkeyform format, -dpassarg	Формат соответственно дополнительных сертификата, закрытого ключа и пассфразы.
-nocert	Если указана эта опция, никакой сертификат не используется. Это обусловливает возможность использования только анонимных шифр-сьютов (сейчас только анонимных DH).
-dhparam filename	Указывает на файл параметров DH, который следует использовать. Эфемерные DH-шифрсьюты создают ключи, использующие набор DH-параметров. Если не указано противного, делается попытка загрузить параметры из файла сертификата сервера. Если это не удается, то будет использован статический набор параметров, жестко встроенный в код команды s_server.
-no_dhe	Если установлена эта опция, никаких DH-параметров загружено не будет, что практически отключает эфемерные DH-шифрсьюты.

Опция	Описание
-verify depth, -Verify depth	Используемая глубина верификации. Опция определяет максимальную длину цепочки сертификатов клиента и заставляет сервер запросить клиентский сертификат. Опция -verify запрашивает сертификат, но клиент не обязан его отправлять, в то время как при указании опции -Verify клиент обязан преодставить сертификат, или произойдет ошибка.
-crl_check, -crl_check_all	Проверяет, что предоставленный клиентом сертификат не был отозван выдавшим его УЦ. Список отозванных сертификатов прикрепляется к файлу сертификата. При использовании опции -crl_check_all проверяются все списки отозванных сертификатов всех УЦ в цепочке.
-CApath directory	Каталог, который следует использовать для проверки клиентского сертификата. Этот каталог должен быть в «хэш-формате», см. раздел 7.6.26 для получения дополнительной информации. Сертификаты из этого каталога также используются для построения цепочки для проверки серверного сертификата.
-CAfile file	Файл, содержащий доверенные сертификаты, которые следует использовать при аутентификации клиента и во время попыток построить цепочку серверных сертификатов. Этот список также используется в списке приемлемых клиентских сертификатов удостоверяющих центров, который передается клиенту при запросе сертификата.
-no_alt_chains	Для более детальной информации см. страницу verify руководства пользователя.
-state	Выводит состояния SSL-сессии.
-debug	Вывести подробную отладочную информацию, включающую шестнадцатеричный дамп всего траффика.
-msg	Вывести все сообщения протокола с шестнадцатеричным дампом.
-nbio_test	Тестирует неблокирующий ввод-вывод
-nbio	Включает неблокирующий ввод-вывод
-crlf	Эта опция переводит символ конца строки с терминала в CR+LF, как требуют некоторые серверы.
-quiet	Подавляет вывод сессионной информации и информации о сертификате.
-psk_hint hint	Использовать подсказку hint идентификатора PSK при использовании шифр-сьюта PSK.
-psk key	Использовать PSK ключ key при использовании шифр-сьюта PSK. Ключ указывается в виде шестнадцатеричного числа без префикса 0x, например, -psk 1a2b3c4d.
-ssl2, -ssl3, -tls1, -tls1_1, -tls1_2, -no_ssl2, -no_ssl3, -no_tls1, -no_tls1_1, -no_tls1_2	Эти опции требуют или запрещают использование указанных протоколов SSL или TLS. По умолчанию при установлении соединения используется гибкий (version-flexible) метод, который позволяет выбрать максимальную взаимно поддерживаемую версию.

Опция	Описание
-bugs	В распространенных реализациях SSL и TLS есть несколько известных ошибок. Указание этой опции включает разнообразные методы их обхода.
-hack	Эта опция включает дальнейшие методы обхода ошибок в некоторых ранних реализациях SSL фирмы Netscape.
-cipher cipherlist	Эта опция позволяет модифицировать список допустимых шифр-сьютов, используемый сервером. Когда клиент отправляет список поддерживаемых шифр-сьютов, используется первый шифр-сьют из списка, включенный в соответствующий список сервера. Поскольку клиент указывает шифр-сьюты в порядке предпочтения, порядок шифрсьютов сервера неважен. Для получения дальнейшей информации см. команду ciphers.
-serverpref	Использовать алгоритм шифрования, предпочтительный для сервера
-tlsextdebug	Печатает содержимое расширений TLS, полученных от сервера, в шестнадцатеричном виде.
-no_ticket	Отключает поддержку сессионных тикетов по RFC4507bis.
-www	Отправляет статусное сообщение клиенту при установлении соединения. Это сообщение включает большое количество информации об используемых шифр-сьютах и различных параметрах сессии. Выводится в HTML-формате, так что эта опция, как правило, используется с веб-браузерами.
-WWW	Эмулирует простой веб-сервер. Страницы будут загружаться относительно текущего каталога, например если запрашивается URL https://myhost/page.html, будет загружен файл ./page.html.
-HTTP	Эмулирует простой веб-сервер. Страницы будут загружаться относительно текущего каталога, например если запрашивается URL https://myhost/page.html, будет загружен файл ./page.html. Предполагается, что загруженные файлы содержат полный и корректный HTML-ответ (строки, являющиеся частью строк и заголовков HTTP-ответа, должны заканчиваться CRLF).
-engine id	выбор энджина (по его уникальной строке id) приводит к тому, что s_server делает попытку получить функциональную ссылку на выбранный энджин, таким образом инициализируя его, если это необходимо. После этого энджин устанавливается, как энджин по умолчанию, для всех доступных алгоритмов.
-id_prefix arg	Создает идентификаторы SSL/TLS сессий, начинающиеся с arg. Это в основном полезно для тестирования любого SSL/TLS-кода (например прокси), который желает иметь дело со множеством серверов, когда каждый из них может генерировать уникальный набор (range) сессионных идентификаторов (например, с определенным префиксом).

Опция	Описание
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-serverinfo file	Файл, содержащий один или более блоков PEM данных. Каждый PEM блок должен содержать закодированное расширение TLS ServerHello (2 байта типа, 2 байта длина, затем следуют байты данных расширения указанной длины). Если клиент отправляет пустое расширение TLS ClientHello указанного типа, то в ответ будет отправлено соответствующее расширение ServerHello.
-no_resumption_on_reneg	Устанавливает флаг SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION
-status	Включает поддержку запроса на статус сертификата (также известный как OCSP stapling)
-status_verbose	Включает поддержку запроса на статус сертификата (также известный как OCSP stapling) и печатает подробный OCSP ответ.
-status_timeout nsec	Устанавливает время ожидания ответа OCSP в nsec секунд.
-status_url url	Устанавливает резервный URL ответчика OCSP, который будет использоваться в том случае, если он отсутствует в серверном сертификате. Без этой опции при отсутствии адреса ответчика в серверном сертификате будет выдано сообщение об ошибке.
-nextprotoneg protocols	Включает TLS расширения Next Protocol Negotiation и позволяет указать список имен протоколов, разделенных знаком запятой. В начале этого списка должны быть наиболее востребованные протоколы. Имена протоколов указываются в кодировке ASCII, например, "http/1.1"или "spdy/3".

7.6.20.4 Команды, используемые при установленном соединении

Если установлен запрос на соединение с SSL-клиентом и не использована опция -www или -WWW, то, как правило, выводятся все данные, полученные от клиента, и все нажатия клавиш будут переданы клиенту.

Также распознаются определенные однобуквенные команды, выполняющие специальные операции. Они перечислены ниже:

q Завершить текущее SSL-соединение, но принимать новые соединения.
Q Завершить текущее SSL-соединение и закончить работу.
r Пересогласовать SSL-сессию.
R Пересогласовать SSL-сессию и запросить клиентский сертификат.
P Отправить некоторый открытый текст по underlying TCP-соединению: это должно заставить клиента прервать соединение из-за нарушения протокола.
S Вывести информацию о статусе кэша сессии.

7.6.20.5 Примечания

Команду s_server можно использовать для отладки SSL-клиентов. Чтобы принять запросы на соединения от веб-браузеров, можно, например, использовать команду

```
openssl s_server -accept 443 -www
```

Хотя указание пустого списка сертификатов удостоверяющих центров при запросе клиентского сертификата, строго говоря, являются нарушением протокола, большинство SSL-клиентов интерпретируют это как то, что приемлемым является сертификат любого удостоверяющего центра. Это полезно для отладочных целей.

Параметры сессии можно вывести с помощью команды sess_id.

7.6.21 Команда `s_time`

7.6.21.1 Описание команды

Команда s_time реализует универсальный SSL/TLS-клиент, который устанавливает соединение с удаленным хостом с помощью SSL/TLS. Она может запросить страницу с сервера и включает время, за которое были переданы полезные данные, в свои изменения времени. Она подсчитывает количество соединений в заданный интервал времени, объем переданных данных (если они есть), и вычисляет среднее время, затраченное на одно соединение.

7.6.21.2 Формат ввода команды

openssl s_time [-connect host:port] [-www page] [-cert filename] [-key filename] [-CApath directory] [-CAfile filename] [-reuse] [-new] [-verify depth] [-nbio] [-time seconds] [-ssl2] [-ssl3] [-bugs] [-cipher cipherlist]

7.6.21.3 Опции команды

Опция	**Описание**
-connect host:port	Указывает хост и опциональный порт для соединения
-www page	Указывает страницу, которую необходимо загрузить с сервера по методу GET. Значение / загружает страницу index.htm[l]. Если этот параметр не указан, команда s_time выполняет только хэндшейк для установки SSL-соединения, но не передает полезные данные.
-cert certname	Используемый сертификат, если его требует сервер. По умолчанию сертификат не используется. Файл должен быть в PEM-формате.
-key keyfile	Используемый закрытый ключ. Если не указан, будет использован закрытый ключ из файла сертификата. Файл должен быть в PEM-формате.

Опция	Описание
-verify depth	Используемая глубина верификации. Эта опция определяет максимальную длину цепочки сертификатов и включает проверку серверных сертификатов. В настоящее время операция верификации продолжается после ошибок, так что все сложности при проверке цепочки сертификатов можно увидеть. Дополнительный эффект: соединение никогда не оборвется из-за неудачной проверки серверного сертификата.
CApath directory	Каталог доверенных сертификатов, используемых при проверке сертификата сервера. Этот каталог должен быть в "хэш-формате". Для более детальной информации смотрите раздел verify. Опция также используется при построении цепочки клиентских сертификатов.
-CAfile file	Файл доверенных сертификатов, используемых при проверке сертификата сервера и попытке построить цепочку клиентского сертификата.
-new	Позволяет использовать при проверке затрачиваемого времени новый ID сессии при каждом соединении. Если не указаны ни опция -new, ни опция -reuse, они обе по умолчанию считаются включенными и выполняются по очереди.
-reuse	Выполняет проверку затрачиваемого времени с использованием одного и того же ID сессии; это можно использовать в качестве проверки того, работает ли кэширование сессии. Если не указаны ни опция -new, ни опция -reuse, они обе по умолчанию считаются включенными и выполняются по очереди.
-nbio	Включает неблокирующий I/O.
-ssl2, -ssl3	Эти опции отключают использование определенных протоколов SSL или TLS. По умолчанию начальный хендшейк использует метод, который должен быть совместим со всеми серверами и позволять им использовать в качестве допустимых протоколы SSL v3, SSL v2 или TLS. Команда для измерения затраченного времени не так богата опциями для включения и выключения протоколов, как команда s_client, и может не подключиться ко всем серверам. К сожалению, существует множество устаревших или испорченных, но действующих серверов, которые не могут справиться с этими методами, и соединение с ними установить не удается. Некоторые серверы работают только в том случае, если протокол TLS отключен с помощью опции -ssl3; другие серверы поддерживают только SSL v2, и им может потребоваться опция -ssl2.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-bugs	В реализациях SSL и TLS есть несколько известных ошибок. Указание этой опции включает различные пути обхода этих ошибок.
-chipher cipherlist	Позволяет модифицировать список шифров, присланных клиентом. Хотя сервер определяет, какой шифрсьют используется, он должен выбрать первый поддерживаемых шифр из списка, присланного клиентом. Для более детальной информации смотри команду cipher.
-time length	Указывает, как долго (в секундах) команда s_time должна устанавливать соединения и опционально — передавать полезные данные с сервера. Скорость работы сервера и клиента, а также скорость связи определяют, сколько соединений может установить команда s_time.

7.6.21.4 Примечания

Можно использовать команду s_time для измерения скорости работы SSL-соединения. Чтобы подключиться к HTTP-серверу по протоколу SSL и получить страницу по умолчанию, обычно используется команда

```
openssl s_time -connect servername:443 -www / -CApath yourdir
-CAfile yourfile.pem -cipher commoncipher [-ssl3]
```

(протокол https использует порт 443). «commoncipher» — это шифр, на который соглашаются и клиент, и сервер.

Если "рукопожатие"выполнить не удается, этому могут быть несколько возможных причин. Если нет ничего столь очевидного, как отсутствие клиентского сертификата, можно попробовать использовать опции -bugs, -ssl2, -ssl3 на случай, если ошибка происходит на сервере. Особенно важно проверить эти опции перед тем, как отправлять сообщение об ошибках в список рассылки OpenSSL.

Частая проблема при попытках заставить работать клиентские сертификаты состоит в том, что веб-клиент жалуется, что у него нет сертификатов, или выдает пустой список выбора. Как правило, это происходит, потому что сервер при запросе сертификата не включает сертификат УЦ, на котором подписан клиентский сертификат, в свой «список допустимых сертификатов УЦ». Этот список можно просмотреть и проверить с помощью команды s_client. Однако, некоторые серверы требуют клиентской аутентификации только после того, как запрошен определенный URL. В этом случае, чтобы получить список, необходимо использовать команду s_client с опцией -prexit и отправить http-запрос на соответствующую страницу.

Если сертификат указан в командной строке с помощью опции -cert, он будет использоваться только в том случае, если сервер явным образом требует клиенский сертификат. Таким образом, простое включение клиентского сертификата в командную строку не дает гарантии, что сертификат будет работать.

7.6.22 Команда `smime`

7.6.22.1 Описание команды

Команда smime обрабатывает почтовые сообщения типа S/MIME. Она может зашифровывать, расшифровывать, подписывать и проверять такие сообщения.

7.6.22.2 Формат ввода команды

openssl smime [-encrypt] [-decrypt] [-sign] [-resign] [-verify] [-pk7out] [-[cipher]]
[-in file] [-no_alt_chains] [-certfile file] [-signer file] [-recip file]
[-inform SMIME|PEM|DER] [-passin arg] [-inkey file] [-out file] [-outform SMIME|PEM|DER]
[-content file] [-to addr] [-from ad] [-subject s] [-text] [-indef] [-noindef]
[-stream] [-rand file(s)] [-md digest] [cert.pem]...

7.6.22.3 Опции команды

Существует шесть операционных опций, которые устанавливают тип производимой операции. Значение остальных опций варьируется в зависимости от типа операции.

Опция	**Описание**
-encrypt	Зашифровывает почту для указанных сертификатов получателей. Входным файлом является незашифрованное сообщение. Выходной файл — зашифрованное почтовое сообщение в формате MIME.
-decrypt	Расшифровывает почту с использованием указанного сертификата и закрытого ключа. В качестве входного файла ожидается зашифрованное почтовое сообщение в формате MIME. В выходной файл записывается расшифрованное сообщение.
-sign	Подписывает почту с использованием указанного сертификата и закрытого ключа. Входным файлом является сообщение, которое необходимо подписать. В выходной файл записывается подписанное сообщение в формате MIME.
-verify	Проверяет подписанную почту. Ожидает подписанное почтовое сообщение в качестве входного файла и выводит подписанные данные. Поддерживаются как незашифрованные, так и зашифрованные подписанные файлы.
-pk7out	Считывает входное сообщение и записывает в выходной файл PKCS#7-структуру в PEM-формате.
-resign	Добавляет к сообщению одну или более новых подписей.
-in filename	Входное сообщение, которое нужно подписать или зашифровать, или сообщение в формате MIME, которое нужно расшифровать или под которым нужно проверить подпись.
-inform SMIME|PEM|DER	Определяет входной формат PKCS#7-структуры. По умолчанию — SMIME, для считывания сообщений в формате S/MIME. Указание на форматы PEM или DER заставляет ожидать в качестве входного файла PKCS#7-структуры в соответствующем формате. Сейчас эта опция влияет только на входной формат PKCS#7-структуры, если никакой PKCS#7-структуры не вводится (например если указаны опции -encrypt или -sign), эта опция игнорируется.

Опция	Описание
-out filename	Текст сообщения, которое было расшифровано или проверено, или выходное сообщение в формате MIME, которое было подписано или проверено.
-outform SMIME|PEM|DER	Определяет выходной формат для структуры PKCS#7. По умолчанию это формат SMIME, который выдает сообщения в формате S/MIME. Если указан параметр PEM и DER, то вместо этого в выводе будут структуры PKCS#7 в формате PEM или DER. В настоящее время это сказывается только на выходном формате структуры PKCS#7; если никакая структура PKCS#7 не выводится (например, при использовании опций -verify или -decrypt), то данная опция никак не влияет на выходной формат.
-stream -indef -noindef	Опции -stream и –indef эквивалентны и разрешают поточный ввод/-вывод для операций кодирования. Это позволяет обрабатывать данные за один проход без необходимости держать в памяти все данные, потенциально поддерживая файлы очень больших размеров. Потоковая обработка автоматически устанавливается для S/MIME подписи с обособленными (detached) данными, если установлен выходной формат SMIME. Для всех остальных операций потоковая обработка по умолчанию отключена.
-noindef	Отключает потоковый ввод/вывод там, где он создал бы сложное кодирование неопределенной длины (ASN.1 constructed encoding). В настоящее время данная опция не работает. В будущем потоковый ввод/вывод будет включаться по умолчанию во всех соответствующих операциях, и эта опция будет его отключать.
-content filename	Определяет файл, содержащий обособленными (detached) данные; полезен только с командой –verify. Используется только, если структура PKCS#7 использует detached-подпись, которая не включает содержимое. Данные, переданные в этой опции, будут использовать вместо любых других, если используется входной формат S/MIME и он использует тип содержимого multipart/signed MIME.
-text	Эта опция добавляет MIME-заголовки text/plain к сообщению при выполнении шифрования или подписи. При дешифровании или проверке подписи эти заголовки удаляются: если расшифрованное или проверенное сообщение не является типом MIME text/plain, то возникает ошибка.
-CAfile file	Файл, содержащий доверенный сертификат удостоверяющего центра. Данная опция используется только с опцией -verify.
-CApath dir	Каталог, содержащий доверенные сертификаты удостоверяющих центров. Используется только с опцией -verify. Этот каталог должен быть стандартным каталогом сертификатов, то есть с каждым сертификатом должна быть связана хэш-сумма поля subject name.
-md digest	Использует алгоритм digest для подписи или переподписи. Если данная опция отсутствует, тогда для генерации ключа подписи будет использоваться алгоритм digest по умолчанию (обычно SHA1).

Опция	Описание
-[cipher]	Использует указанный алгоритм шифрования. Например, DES (56 бит) – это –des, тройной DES (168 бит) - -des3, также может использоваться функция EVP_get_cipherbyname() с начальным символом тире, например, -aes_128_cbc. Чтобы узнать список поддерживаемых шифров в вашей версии OpenSSL смотри enc. Если опция не указана, то используется тройной DES. Используется только с опцией -encrypt.
-nointern	При проверке подписи, как правило, сертификат отправителя ищется среди сертификатов, включенных в сообщение (если таковые есть). Если указана данная опция, используются только сертификаты, указанные в опции -certfile. Сертификаты, включенные в сообщение, могут использоваться в качестве недоверенных сертификатов удостоверяющих центров.
-noverify	Не проверять сертификат отправителя подписанного сообщения.
-nochain	Не выполнять проверку цепочки доверия сертификатов отправителя, то есть не использовать сертификаты, включенные в подписанное сообщение, в качестве недоверенных сертификатов удостоверяющего центра.
-nosigs	Не пытаться проверять подписи под сообщением.
-nocerts	При подписывании сообщения сертификат отправителя, как правило, включается в сообщение. При указании данной опции сертификат отправителя в сообщение не включается. Это уменьшает размер подписанного сообщения, но получатель должен иметь на своем компьютере копию сертификата отправителя (например, переданную с помощью опции -certfile).
-noattr	Обычно, когда сообщение подписано, то добавляется набор атрибутов, включающий время подписи и поддерживаемые симметричные алгоритмы. При использовании данной опции атрибуты не добавляются.
-binary	Как правило, входное сообщение переводится в «канонический» формат, использующий CR и LF в качестве концов строк, как требует спецификация S/MIME. При указании данной опции перевода в такой формат не производится. Это полезно при передаче бинарных данных, которые передаются не в MIME-формате.
-nodetach	«Непрозрачное» подписание сообщения: эта форма более устойчива при почтовой передаче, но ее не смогут прочитать почтовые программы, не поддерживающие формат S/MIME. Если эта опция не указана, выполняется «прозрачное» подписание с использованием типа MIME multipart/signed.
-certfile file	Позволяет указать дополнительные сертификаты. При подписывании сообщения эти сертификаты будут включены в сообщение. При проверке сообщения среди этих сертификатов будет искаться сертификат отправителя. Сертификаты должны быть в PEM-формате.
-signer file	Сертификат отправителя при подписи сообщения. При проверке сообщения сертификаты отправителя будут записаны в этот файл, если проверка была успешной.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-recip file	Сертификат получателя при расшифровании сообщения. Этот сертификат должен принадлежать одному из получателей сообщения, иначе выводится сообщение об ошибке.
-inkey file	Закрытый ключ, который следует использовать при подписывании или расшифровании сообщения. Закрытый ключ должен соответствовать сертификату. Если эта опция не указана, закрытый ключ должен быть включен в файл сертификата, указанный в опции -recip или -signer.
-passin arg	Источник пароля для закрытого ключа. Для получения дополнительной информации о формате аргумента arg см. раздел 7.4.
-rand file(s)	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел, или сокет EGD. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
cert.pem...	Один или больше сертификатов получателей. Используется при зашифровании сообщения.
-to, -from, -subject	Соответствующие заголовки почтовых сообщений. Они включаются снаружи подписанной части сообщения, чтобы их можно было включить вручную. Многие почтовые клиенты, работающие с форматом S/MIME, проверяют, совпадает ли почтовый адрес, указанный в сертификате отправителя, с почтовым адресом отправителя.
-purpose, -ignore_critical, -issuer_checks, -crl_check, -crl_check_all, -policy_check, -extended_crl, -x509_strict, -policy, -check_ss_sig, -no_alt_chains	Устанавливают различные опции при проверке цепочки сертификатов. Для более детальной информации см. страницу verify руководства пользователя.
-gost89	Используемый алгоритм зашифрования. Используется только с опцией -encrypt. При шифровании с помощью алгоритмов ГОСТ данная опция является обязательной.

7.6.22.4 Примечания

Заголовки MIME-сообщения при отправке не должны отделяться от остального содержания пустыми строками. Некоторые почтовые программы автоматически добавляют такие пустые строки. Направить почту непосредственно в программу sendmail — один из способов получить корректный формат.

Подписываемое и зашифрованное сообщение должно включать необходимые MIME-заголовки, иначе многие почтовые клиенты не смогут его корректно воспроизвести (или во-

обще не смогут). Вы можете использовать опцию -text для автоматического добавления заголовков.

«Подписанное и зашифрованное» сообщение — сообщение, сначала подписанное, затем зашифрованное. Такое сообщение можно получить, зашифровав уже подписанное сообщение (см. раздел 7.6.22.6).

Оригинальная версия не поддерживает возможность создания нескольких подписей под одним почтовым сообщением, но может проверять корректность сообщений с несколькими подписями.

Некоторые клиенты S/MIME неправильно работают, если сообщение содержит несколько подписей. Можно подписывать сообщения "параллельно когда подписывается уже подписанное сообщение.

Функциональность создания второй и последующих подписей (опция -add) добавлена в МагПро КриптоПакет.

Опции -encrypt и -decrypt отражают обычное использование соответствующих функций в почтовых клиентах. Строго говоря, эти опции работают с разновидностью enveloped data формата PKCS#7. PKCS#7 encrypted data используется для других целей.

Опция -resign использует существующий хэш сообщения, когда добавляет новую подпись. Это означает, что сообщение уже должно содержать как минимум одну подпись, использующую тот же самый алгоритм хэширования. В противном случае операция не будет выполнена.

Опции -stream и -indef включают экспериментальную поддержку поточного ввода-вывода. В результате получается данные выводятся в не в кодировке DER, а в кодировке BER с использованием составного кодирования неопределенной длины. Поточный ввод-вывод поддерживается для операции -encrypt, для операции -sign - в том случае, если подпись не отделяется.

Поточный ввод-вывод всегда используется для команды -sign с обособленными данными, но поскольку контент больше не является частью структуры PKCS#7, используется кодировка DER.

7.6.22.5 Коды выхода

1 Операция выполнена полностью успешно.
2 Ошибка при обработке опций команды.
3 Один из входных файлов не прочитан.
4 Ошибка при создании PKCS#7-файла или считывании MIME-сообщения.
5 При проверке сообщение признано корректным, но произошла ошибка при записи одного из сертификатов отправителя.

7.6.22.6 Примеры

Создать «прозрачно» подписанное сообщение:

```
openssl smime -sign -in message.txt -text -out mail.msg -signer
mycert.pem
```

Создать «непрозрачно» подписанное сообщение:

```
openssl smime -sign -in message.txt -text -out mail.msg -nodetach
-signer mycert.pem
```

Создать подписанное сообщение, включить несколько дополнительных сертификатов и считать закрытый ключ из другого файла:

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

```
openssl smime -sign -in in.txt -text -out mail.msg -signer
mycert.pem -inkey mykey.pem -certfile mycerts.pem
```

Отправить подписанное сообщение в Unix-подобными ОС прямо в программу sendmail, включая заголовки:

```
openssl smime -sign -in in.txt -text -signer mycert.pem -from
steve@openssl.org -to someone@somewhere -subject ''Signed message''
| sendmail someone@somewhere
```

Проверить сообщение и в случае успешной проверки сохранить сертификат отправителя в файле:

```
openssl smime -verify -in mail.msg -signer user.pem -out
signedtext.txt
```

Отправить зашифрованное сообщение, используя алгоритм gost89:

```
openssl smime -encrypt -in in.txt -from steve@openssl.org -to
someone@somewhere -subject <<Encrypted message>> -gost89 user.pem -out
mail.msg
```

Подписать и зашифровать сообщение:

```
openssl smime -sign -in ml.txt -signer my.pem -text | openssl smime
-encrypt -out mail.msg -from steve@openssl.org -to someone@somewhere
-subject ''Signed and Encrypted message'' -gost89 user.pem
```

Примечание: команда зашифрования не включает опцию -text, потому что зашифровываемое сообщение уже включает MIME-заголовки.

Расшифровать сообщение:

```
openssl smime -decrypt -in mail.msg -recip mycert.pem -inkey key.pem
```

Выходными данными из подписывающей программы Netscape является PKCS#7-структура в формате с отдельной подписью. Вы можете использовать эту программу, чтобы проверить такую подпись, разбив на строки структуру в кодировке base64, окружив ее ограничителями:

```
-----BEGIN PKCS7-----
-----END PKCS7-----
```

и воспользовавшись командой

```
openssl smime -verify -inform PEM -in signature.pem -content
content.txt
```

Вы также можете декодировать подпись из кодировки base64 и воспользоваться командой

```
openssl smime -verify -inform DER -in signature.der -content
content.txt
```

7.6.23 Команда `speed`

7.6.23.1 Описание команды

Эта команда используется, чтобы тестировать скорость криптографических алгоритмов.

7.6.23.2 Формат ввода команды

openssl speed [-engine id] -evp alg

7.6.23.3 Опции команды

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-engine id	При указании модуля engine (по его уникальной строке id) команда speed попытается получить функциональную ссылку на указанный моудль, инициализируя его, если необходимо. Затем этот модуль будет установлен как умолчательный для всех имеющихся алгоритмов.
-evp alg	Измеряет скорость работы алгоритмов зашифрования (gost89, gost89-cnt, gost89-cnt-12, gost89-cbc) или хэширования (md_gost12_512, md_gost12_256, md_gost94).

7.6.24 Команда `spkac`

7.6.24.1 Описание команды

Команда spkac обрабатывает файлы Netscape SPKAC, содержащие подписанный открытый ключ и запрос. Команда может печатать содержимое этих файлов, проверять подпись и создавать свои собственные файлы SPKAC по имеющемуся закрытому ключу.

7.6.24.2 Формат вызова команды

openssl spkac [-in filename] [-out filename] [-key keyfile] [-passin arg] [-challenge string] [-pubkey] [-spkac spkacname] [-spksect section] [-noout] [-verify] [-engine id]

7.6.24.3 Опции команды

Опция	Описание
-in filename	Определяет имя входного файла для чтения из него информации; если данный параметр не указан, то используется стандартный ввод. Опция игнорируется при использовании опции –key.
-out filename	Определяет имя выходного файла для записи; по умолчанию используется стандартный вывод.
-key keyfile	Создает файл SPKAC, используя закрытый ключ из keyfile. Если используется вместе с опциями -in, -noout, -spksect и –verify, то последние игнорируются.
-passin password	Источник пароля для входного файла. Для более детальной информации о формате arg смотрите раздел Пароли как аргументы.
-challenge string	Определяет строку запроса при создании SPKAC.
-spkac spkacname	Позволяет указать альтернативное имя формы переменной, содержащей SPKAC. По умолчанию это "SPKAC". Эта опция влияет и на создаваемые, и на входные SPKAC файлы.
-spksect section	Позволяет указывать альтернативное имя формы секции, содержащей SPKAC. Если опция не указана, то используется секция по умолчанию.
-noout	Не выводить текстовую версию SPKAC (не используется при создании SPKAC)

Опция	Описание
-pubkey	Выводит открытый ключ в формате SPKAC (не используется при создании SPKAC)
-verify	Проверить электронную подпись в указанном SPKAC.
-engine id	Указание модуля engine (по его уникальной идентификационной строке) заставит команду spkac попытаться получить функциональную ссылку на указанный модуль, инициализируя его, если необходимо. Затем модуль engine будет установлен как умолчательный для всех доступных алгоритмов.

7.6.24.4 Примечания

Созданный SPKAC с соответствующими добавленными DN компонентами может использоваться в утилите ca. SPKAC обычно генерируется Netscape при отправке формы, содержащей метку KEYGEN, как часть процесса выпуска сертификата. Строка запроса позволяет использовать примитивную форму доказательства владения закрытым ключом. Проверка подписи SPKAC и наличие случайной строки запроса является определенной гарантией, что пользователю известен закрытый ключ, соответствующий открытому ключу в сертификате. Это важно в некоторых приложениях. Без этого предыдущий SPKAC может быть использован для атаки типа "replay attack".

7.6.24.5 Примеры

Распечатать содержимое SPKAC:

```
openssl spkac -in spkac.cnf
```

Проверить подпись SPKAC:

```
openssl spkac -in spkac.cnf -noout -verify
```

Создать SPKAC, используя строку запроса "hello":

```
openssl spkac -key key.pem -challenge hello -out spkac.cnf
```

Пример SPKAC (длинная строка разделена для удобства просмотра)

```
SPKAC=MIG5MGUwXDANBgkqhkiG9w0BAQEFAANLADBIAkEA1cCoq2Wa3Ixs47uI7F\
PVwHVIPDx5yso105Y6zpozam135a8R0CpoRvkkigIyXfcCjiVi5oWk+6FfPaD03u\
PFoQIDAQABFgVoZWxsbzANBgkqhkiG9w0BAQQFAANBAFpQtY/FojdwkJh1bEIYuc\
2EeM2KHTWPEepWYeawvHD0gQ3DngSC75YCWnnDdq+NQ3F+X4deMx9AaEglZtULwV\
4=
```

7.6.25 Команда ts

7.6.25.1 Описание команды

Команда ts — это базовое клиент-серверное приложение службы меток времени (CMB), соответствующее RFC 3161. Служба меток времени может быть частью реализации PKI. Ее задача — предоставлять долгосрочные свидетельства существования определенных данных до определенного времени. Вот краткое описание протокола:

1. Клиент CMB вычисляет значение однонаправленной хэш-функции для файла данных и отправляет хэш в CMB.

2. CMB присоединяет текущую дату и время к полученному хэшу, подписывает все вместе и отправляет полученный маркер метки времени обратно клиенту. Созданием этого маркера CMB удостоверяет существование исходного файла данных во время создания ответа.

3. Клиент CMB получает ответ метки времени и проверяет подпись под ним. Кроме того, клиент проверяет, совпадает ли значение хэша, содержащееся в ответе, со значением, отправленным в CMB.

Существует один блок DER-кодированных данных протокола, определенный для транспортирования метки времени в CMB, и один для отправки ответа метки времени обратно клиенту. Команда ts имеет три основные функции: создание запроса на метку времени на основании файла данных, создание маркера метки времени на основании запроса и проверка того, соответствует ли маркер конкретному запросу или файлу данных.

Пока не поддерживается автоматическая отправка запросов/маркеров по HTTP или TCP, как предлагается в RFC 3161. Пользователи должны отправлять запросы по ftp или электронной почте.

7.6.25.2 Формат ввода команды

openssl ts -query [-rand file:file...] [-config configfile] [-data file_to_hash] [-digest digest_bytes] [-md_gost12_256|-md_gost12_512|-md_gost94|-md2|-md4|-md5|-sha|-sha1|-mdc2|-ripemd160|...] [-policy object_id] [-no_nonce] [-cert] [-in request.tsq] [-out request.tsq] [-text]

openssl ts -reply [-config configfile] [-section tsa_section] [-query- file request.tsq] [-passin arg] [-signer tsa_cert.pem] [-inkey private.pem] [-chain certs_file.pem] [-policy object_id] [-in response.tsr] [-token_in] [-out response.tsr] [-token_out] [-text] [-engine id]

openssl ts -verify [-data file_to_hash] [-digest digest_bytes] [-query- file request.tsq] [-in response.tsr] [-token_in] [-CApath trusted_cert_path] [-CAfile trusted_certs.pem] [-untrusted cert_file.pem]

7.6.25.3 Опции команды

7.6.25.3.1 Создание запроса на метку времени

Можно использовать опцию -query для создания и вывода запроса на метку времени со следующими опциями:

Опция	**Описание**
-rand file:file...	Файл или файлы, содержащие случайные данные, используемые в качестве начальных данных для генератора случайных чисел. Можно указывать несколько файлов, разделяя их символами, соответствующими конкретной ОС. Разделитель точка с запятой (;) – для MS-Windows, запятая (,) – для OpenVMS, и двоеточие (:) – для всех остальных ОС.
-config configfile	Файл конфигурации, который следует использовать. Эта опция имеет преимущество перед переменной среды OPENSSL_CONF. С опцией -query используется только секция OID конфигурационного файла. (Необязательная опция)
-data file_to_hash	Файл данных, для которого нужно создать запрос на метку времени. Если не указан ни параметр -data, ни параметр -digest, по умолчанию используется стандартный вход. (Необязательная опция)
-digest digest_bytes	Можно указать непосредственно хэш сообщения, не указывая файл данных. Хэш должен быть указан в шестнадцатеричном формате, два знака на байт, байты могут быть разделены двоеточиями (например, 1A:F6:01:... или 1AF601...). Количество байтов должно соответствовать использоваемому алгоритму хэш-функции. (Необязательная опция)
-md_gost12_256|-md_gost12_512|-md_gost94|-md2|-md4|-md5|-sha|-sha1|-mdc2|-ripemd160|...	Алгоритм хэширования, который будет применяться к файлу данных; поддерживаются все алгоритмы, поддерживаемые командой openssl dgst. По умолчанию используется SHA-1, поэтому при работе с алгоритмами ГОСТ данную опцию указывать необходимо.
-policy object_id	Регламент CMB, в соответствии с которым клиент ожидает создания ответа меток времени. Можно использовать или нотацию OID с точками, или наименования OID, определенные в конфигурационном файле. Если никакой регламент не запрошен, CMB использует свой умолчательный регламент. (Необязательная опция)

Опция	Описание
-no_nonce	Если эта опция указана, в запросе не указывается расширение nonce. В противном случае в запрос включается 64-битное псевдослучайное расширение nonce. Рекомендуется использовать nonce для защиты против взлома путем замещения оригинала. (Необязательная опция)
-cert	Клиент ожидает, что CMB включит в ответ сертификат подписи. (Необязательная опция)
-in request.tsq	Эта опция указывает созданный ранее запрос на метку времени в кодировке DER, который будет записан в выходной файл. Полезно, если необходимо просмотреть содержимое запроса в формате открытого текста. (Необязательная опция)
-out request.tsq	Имя выходного файла, в который записывается запрос. По умолчанию — стандартный вывод. (Необязательная опция)
-text	Если эта опция указана, в выходной файл выводятся данные не в DER-кодировке, а в формате открытого текста. (Необязательная опция)

7.6.25.3.2 Создание ответа метки времени

Ответ метки времени (TimeStampResp) состоит из статуса ответа и собственно маркера метки времени (ContentInfo), если таковой успешно создан. Для создания ответа метки времени или маркера метки времени на основании запроса и вывода ответа/маркера открытым текстом используется команда -reply. Если не указана опция -token_out, результатом работы данной опции всегда является ответ метки времени (TimeStampResp), в противном случае — маркер метки времени (ContentInfo).

Опция	Описание
-config configfile	Файл конфигурации, который следует использовать. Эта опция имеет преимущество перед переменной среды OPENSSL_CONF. Описание конфигурируемых переменных см. в разделе 7.6.25.4. (Необязательная опция)
-section tsa_section	Наименование секции конфигурационного файла, содержащей установки для создания ответа. Если эта опция не указана, используется умолчательная секция CMB (подробности см. в разделе 7.6.25.4). (Необязательная опция)
-queryfile request.tsq	Имя файла, содержащего DER-закодированный запрос на метку времени. (Необязательная опция)

Опция	Описание
-passin password_src	Указывает, где содержится пароль для закрытого ключа. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4. (Необязательная опция)
-signer tsa_cert.pem	Сертификат подписи CMD в PEM-кодировке. Сертификат подписи CMB должен содержать ровно одно расширение extended key usage, связанное с ним, а именно timeStamping. Расширение extended key usage также должно быть критическим, иначе сертификат будет отвергнут. Имеет преимущество над переменной конфигурационного файла signer_cert. (Необязательная опция)
-inkey private.pem	Закрытый ключ подписи CMD в кодировке PEM. Имеет преимущество над переменной конфигурационного файла signer_key. (Необязательная опция)
-chain certs_file.pem	Набор сертификатов в PEM-кодировке, который будет полностью включен в ответ вместе с сертификатом подписи CMB, если запрос создавался с опцией -cert. Предполагается, что этот файл содержит цепочку сертификатов для проверки сертификата подписи CMB, от сертификата УЦ, выпустившего сертификат подписи CMB, и выше. Опция -reply не создает цепочку сертификатов автоматически. (Необязательная опция)
-policy object_id	Умолчательный регламент создания ответа, если клиент не указал явным образом конкретный регламент CMB. OID может быть указан с помощью точечной нотации или имени. Имеет преимущество перед опцией конфигурационного файла default_policy. (Необязательная опция)
-in response.tsr	Указывает ранее созданный ответ метки времени или маркер метки времени (если указана также опция -token_in) в DER-кодировке, который будет записан в выходной файл. Эта опция не требует запроса, она полезна, например, если необходимо просмотреть содержание ответа или маркера или если нужно извлечь маркер метки времени из ответа. Если входным файлом является маркер, а выходным - ответ метки времени, к маркеру добавляется умолчательный «гарантированный» статус. (Необязательная опция)
-token_in	Этот флаг может быть использован вместе с опцией -in и указывает, что входными данными является маркер метки времени (ContentInfo), а не ответ метки времени (TimeStampResp). (Необязательная опция)

Опция	Описание
-out response.tsr	Ответ записывается в этот файл. Формат и содержание файла зависит от других опций (см. -text, -token_out). Умолчание - стандартный выход. (Необязательная опция)
-token_out	Выходными данными является маркер метки времени (ContentInfo), а не ответ метки времени (TimeStampResp). (Необязательная опция)
-text	Если эта опция указана, вывод производится открытым текстом, а не в DER-кодировке. (Необязательная опция)
-engine id	Указание энджина(по его уникальной идентификационной строке) заставляет команду ts попытаться получить функциональную ссылку на указанный модуль, при необходимости инициализируя его. Затем энджин будет установлен в качестве умолчательного для всех доступных алгоритмов. Умолчание — builtin. (Необязательная опция)

7.6.25.3.3 Проверка ответа метки времени

Опция -verify используется для проверки того, действителен ли ответ метки времени или маркер метки времени, и соответствует ли он конкретному запросу метки времени или файлу данных. Опция -verify не использует файл конфигурации.

Опция	Описание
-data file_to_hash	Следует проверить, соответствует ли ответ или маркер файлу file_to_hash. Файл хэшируется алгоритмом хэш-функции, указанном в маркере. Если эта опция указана, нельзя указывать опции -digest и -queryfile. (Необязательная опция)
-digest digest_bytes	Следует проверить, соответствует ли ответ или маркер хэшу, указанному в этой опции. Количество байтов должно соответствовать алгоритму хэш-функции, указанному в маркере. Если эта опция указана, нельзя указывать опции -digest и -queryfile. (Необязательная опция)
-queryfile request.tsq	Исходный запрос метки времени в DER-кодировке. Если эта опция указана, нельзя указывать опции -data и -digest. (Необязательная опция)
-in response.tsr	Ответ метки времени, который необходимо проверить, в DER-кодировке. (Обязательная опция)

Опция	Описание
-token_in	Этот флаг может быть использован вместе с опцией -in и указывает, что входными данными является маркер метки времени в кодировке DER (ContentInfo), а не ответ метки времени (TimeStampResp). (Необязательная опция)
-CApath trusted_cert_path	Наименование каталога, содержащего доверенные сертификаты корневых удостоверяющих центров клиента. Для получения дополнительной информации см. похожую опцию команды verify (см. раздел 7.6.26). Необходимо указать либо эту опцию, либо опцию -CAfile. (Необязательная опция)
-CAfile trusted_certs.pem	Наименование файла, содержащего набор доверенных самоподписанных сертификатов удостоверяющих центров в PEM-кодировке. Для получения дополнительной информации см. похожую опцию команды verify (см. раздел 7.6.26). Необходимо указать либо эту опцию, либо опцию -CApath. (Необязательная опция)
-untrusted cert_file.pem	Набор дополнительных недоверенных сертификатов в PEM-кодировке, которые могут понадобиться для создания цепочки сертификатов для проверки сертификата подписи CMB. Этот файл должен содержать сертификат подписи CMB и все сертификаты промежуточных удостоверяющих центров, если только они уже не включены в ответ.

7.6.25.4 Опции конфигурационного файла

Опции -query и -reply используют конфигурационный файл, определенный переменной среды OPENSSL_CONF. Опция -query использует только секцию символическиех имен OID и может работать и без нее. Но опции -reply для работы конфигурационный файл необходим.

Если в командной строке присутствует опция, эквивалентная переменной конфигурационного файла, опция командной строки всегда имеет преимущество над установками конфигурационного файла.

Опция	Описание
tsa section, default_tsa	Это главная секция, она определяет имя другой секции, содержащей все опции, сопутствующие опции -reply. Эта умолчательная секция может быть переуказана опцией командной строки -section. (Необязательная опция)
oid_file	Описание см. в разделе 7.6.2. (Необязательная опция)
oid_section	Описание см. в разделе 7.6.2. (Необязательная опция)
RANDFILE	Описание см. в разделе 7.6.2. (Необязательная опция)

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
serial	Имя файла, содержащего шестнадцатеричный серийный номер последнего созданного ответа метки времени. Этот номер с каждым ответом увеличивается на 1. Если во время создания ответа такой файл не существует, создается новый файл с серийным номером 1. (Обязательная опция)
crypto_device	Указывает модуль engine библиотеки OpenSSL, который будет установлен в качестве умолчательного для всех доступных алгоритмов. Умолчательное значение — builtin, можно указать любые другие модули, поддерживаемые OpenSSL (например, chil или the Ncipher HSM). (Необязательная опция)
signer_cert	Сертификат подписи CMB в PEM-кодировке. То же, что опция командной строки -signer. (Необязательная опция)
certs	Файл, содержащий набор сертификатов в PEM-кодировке, которые необходимо включить в ответ. То же, что опция командной строки -chain. (Необязательная опция)
signer_key	Закрытый ключ CMB в PEM-кодировке. То же, что опция командной строки -inkey. (Необязательная опция)
default_policy	Регламент для использования по умолчанию, когда запрос не указывает никаких регламентов. То же, что опция командной строки -policy. (Необязательная опция)
other_policies	Список регламентов, разделенных запятой, которые также могут быть приняты CMB, и используются только в том случае, если запрос прямо указывает один из них. (Необязательная опция)
digests	Список алгоритмов дайджеста, которые принимает CMB. Необходимо указать по крайней мере один алгоритм. (Обязательная опция)
accuracy	Точность источника времени CMB в секундах, миллисекундах и микросекундах. Например, secs:1, millisecs:500, microsecs:100. Если один из компонентов отсутствует, соответствующее поле заполняется нулевым значением. (Необязательная опция)
clock_precision_digits	Указывает максимальное количество цифр, представляющих долю секунды, которые необходимо включить в метку времени. Нули справа от последней значащей цифры следует удалять из указания времени, поэтому там на самом деле может оказаться меньше цифр, или даже вообще не указаны доли секунды. Максимальное значение — 6, умолчательное — 0. (Необязательная опция)

Опция	Описание
ordering	Если значение этой опции yes, ответы, созданные этой CMB, всегда можно заказать, даже если временное различие между двумя ответами меньше суммы их точностей. Умолчательное значение — no. (Необязательная опция)
tsa_name	Значение этой опции следует устанавливать в yes, если необходимо включать subject name сертификата CMB в поле TSA name (наименование CMB) в ответ. Умолчательное значение — no. (Необязательная опция)
ess_cert_id_chain	Объекты SignedData, созданные CMB, всегда содержат идентификатор сертификата подписи в подписанном атрибуте (см. RFC 2634.) Если значение этой опции установлено в yes, и если указана или переменная certs, или опция -chain, то идентификаторы сертификатов цепи также будут включены в подписанный атрибут SigningCertificate. Если значение этой переменной установлено в no, то включается только идентификатор сертификата подписи. Умолчательное значение — no. (Необязательная опция)

7.6.25.5 Переменные среды

Переменная среды OPENSSL_CONF содержит путь к конфигурационному файлу. Если указана опция командной строки -config, эта опция имеет преимущество над OPENSSL_CONF.

7.6.25.6 Примеры

Все примеры, приведенные в данном разделе, предполагают, что переменная среды OPENSSL_CONF указывает на действительный конфигурационный файл, например, в этом качестве подходит типовой конфигурационный файл openssl/apps/openssl.cnf.

7.6.25.6.1 Запрос метки времени

Создать запрос на метку времени для design1.txt с помощью SHA-1 без расширения nonce, без указания регламента и без включения сертификатов в ответ:

```
openssl ts -query -data design1.txt -no_nonce -out design1.tsq
```

Создать подобный запрос на метку времени, явно указывая дайджест сообщения:

```
openssl ts -query -digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b
-no_nonce -out design1.tsq
```

Вывести содержимое предыдущего запроса открытым текстом:

```
openssl ts -query -in design1.tsq -text
```

Создать запрос метки времени, включающий дайджест MD-5 файла design2.txt, с запросом сертификата подписи и расширения nonce, указанием идентификатора регламента (считается, что наименование tsa_policy1 указано в секции OID конфигурационного файла):

```
openssl ts -query -data design2.txt -md5 -policy tsa_policy1
-cert -out design2.tsq
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.25.6.2 Ответ метки времени

Перед созданием ответа метки времени следует создать сертификат подписи CMB, содержащий критическое расширение extended key usage со значением timeStamping и без каких-либо других расширений key usage. Чтобы создать соответствующий сертификат, можно добавить строку *extendedKeyUsage = critical,timeStamping* в секцию пользовательских сертификатов в конфигурационном файле. О создании сертификата см. разделы 7.6.18, 7.6.2, 7.6.28. Приведенные в данном разделе примеры предполагают, что сертификат УЦ содержится в файле cacert.pem, сертификат подписи, подписанный на cacert.pem — в файле tsacert.pem, закрытый ключ CMB — в файле tsakey.pem.

Создать ответ метки времени на запрос:

```
openssl ts -reply -queryfile design1.tsq -inkey tsakey.pem \
 -signer tsacert.pem -out design1.tsr
```

Если использовать установки конфигурационного файла, можно написать только:

```
openssl ts -reply -queryfile design1.tsq -out design1.tsr
```

Распечатать ответ метки времени на стандартный выход в удобочитаемом формате:

```
openssl ts -reply -in design1.tsr -text
```

Создать маркер метки времени вместо ответа метки времени:

```
openssl ts -reply -queryfile design1.tsq -out design1_token.der -token_out
```

Распечатать маркер метки времени на стандартный выход в удобочитаемом формате:

```
openssl ts -reply -in design1_token.der -token_in -text -token_out
```

Извлечь маркер метки времени из ответа:

```
openssl ts -reply -in design1.tsr -out design1_token.der -token_out
```

Добавить информацию о статусе "выдано"к маркеру метки времени, создав таким образом корректный ответ:

```
openssl ts -reply -in design1_token.der -token_in -out design1.tsr
```

7.6.25.6.3 Проверка метки времени

Проверить, соответствует ли ответ метки времени запросу:

```
openssl ts -verify -queryfile design1.tsq -in design1.tsr
-CAfile cacert.pem -untrusted tsacert.pem
```

Проверить ответ метки времени, содержащий цепочку сертификатов:

```
openssl ts -verify -queryfile design2.tsq -in design2.tsr
-CAfile cacert.pem
```

Проверить, соответствует ли маркер метки времени исходному файлу данных:

```
openssl ts -verify -data design2.txt -in design2.tsr -CAfile cacert.pem
```

Проверить, соответствует ли маркер метки времени дайджесту сообщения:

```
openssl ts -verify -digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b
-in design2.tsr -CAfile cacert.pem
```

Дополнительные примеры можно найти также в каталоге test.

7.6.26 Команда `verify`

7.6.26.1 Описание команды

Команда verify проверяет цепочки сертификатов.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.26.2 Формат ввода команды

openssl verify [-CApath directory] [-CAfile file] [-purpose purpose] [-policy arg] [-ignore_critical] [-attime timestamp] [-check_ss_sig] [-crlfile file] [-crl_download] [-crl_check] [-crl_check_all] [-policy_check] [-explicit_policy] [-inhibit_any] [-inhibit_map] [-x509_strict] [-extended_crl] [-use_deltas] [-policy_print] [-no_alt_chains] [-untrusted file] [-help] [-issuer_checks] [-trusted file] [-verbose] [-] [certificates]

7.6.26.3 Опции команды

Опция	**Описание**
-CApath directory	Каталог доверенных сертификатов. Сертификаты должны иметь имена формата: hash.0 или иметь символические ссылки на них в таком же формате (здесь «hash» — хэшированное значение поля subject name; см. раздел 7.6.28.) Под Unix-подобными ОС скрипт c_rehash автоматически создает символические ссылки на каталог сертификатов.
-CAfile file	Файл доверенных сертификатов. Файл должен содержать больше одного сертификата в PEM-формате, конкатенированных вместе.
-attime timestamp	Выполняет проверку, используя время, указанное в timestamp, а не текущее системное время. timestamp – это число, указывающая количество секунд, начиная с 01.01.1970 (время UNIX).
-check_ss_sig	Выполняет проверку подписей под самоподписанными сертификатами. По умолчанию такая проверка не выполняется, так как от подмены самоподписанного сертификата она не защищает.
-crlfile file	Файл, содержащий один или более список отзыва сертификатов (в формате PEM), для загрузки.
-crl_download	Производит попытку загрузки CRL (списка отозванных сертификатов) для данного сертификата.
-crl_check	Выполняет проверку на наличие сертификата подписи в соответствующем списке отзыва.
-untrusted file	Файл недоверенных сертификатов. Этот файл должен содержать больше одного сертификата
-purpose purpose	Назначение сертификата. Без этой опции никакой цепочечной проверки не выполняется. В настоящее время возможны следующие значения этой опции: sslclient, sslserver, nssslserver, smimesign, smimeencrypt. Для получения дополнительной информации см. раздел 7.6.26.4.
-help	Выводит сообщение об использовании.
-verbose	Выводит дополнительную информацию о выполняемых операциях.
-issuer_checks	Выводит диагностику, связанную с поиском сертификата, на котором заверен обрабатываемый сертификат. Эта диагностика показывает, почему каждый из рассмотренных сертификатов отвергнут. Однако само по себе присутствие сообщений об отказе не означает, что что-то не так — во время обычного процесса проверки может произойти несколько отказов.

Опция	Описание
-policy arg	Включает обработку политик и добавить arg в user-initial-policy-set (см. RFC5280). Политика arg может быть именем объекта или OID в цифровой форме. Этот аргумент может указываться несколько раз.
-policy_check	Включает проверку соответствия сертификатов политике указанной в сертификате удостоверяющего центра
-explicit_policy	Требует явного указания политики
-inhibit_any	Устанавливает переменную policy в значение inhibit-any-policy (см. RFC 3280 и др.).
-inhibit_map	Устанавливает переменную policy в значение inhibit-policy-mapping (см. RFC 3280 и др.).
-no_alt_chains	При построении цепочки сертификатов если первая построенная цепочка оказывается недоверенной, то OpenSSL будет продолжать поиск альтернативной цепочки доверия. С этой опцией такое поведение подавляется таким образом, что проверяется только первая найденная цепочка. Использование данной опции приведет к тому, что поведение OpenSSL будет соответствовать его более старым версиям.
-trusted file	Файл дополнительных доверенных сертификатов. Файл должен содержать несколько сертификатов в PEM формате, соединенные вместе.
-policy_print	Выводит диагностику, связанную с проверкой policy
-crl_check	Выполняет проверку на наличие сертификата подписи в соответствующем списке отзыва
-crl_check_all	Выполняет проверку на наличие соответствующем списке отзыва всех сертификатов из цепочки доверия
-ignore_critical	Игнорирует при проверке сертификата неизвестные расширения X509v3, помеченные как критичные.
-x509_strict	Строгая проверка соответствия сертификатов формату x509
-extended_crl	Позволяет использовать дополнительные возможности списков отзыва сертификатов, такие как косвенные списки отзыва и альтернативные ключи подписи списков отзыва.
-use_deltas	Включает поддержку дельта-списков отзыва сертификатов.
-check_ss_sig	Выполняет проверку подписей под самоподписанными сертификатами. По умолчанию такая проверка не выполняется, так как от подмены самоподписанного сертификата она не защищает.
-	Отмечает последнюю опцию. Все аргументы, следующие за этой опцией, считаются файлами сертификатов. Это полезно, если имя первого файла сертификатов начинается с ”-”.
certificates	Один или больше сертификатов, которые необходимо проверить. Если не указано ни одного имени сертификата, делается попытка считать сертификат со стандартного входа. Все сертификаты должны быть в формате PEM.

7.6.26.4 Операция проверки

Команда verify использует те же функции, что и internal SSL and S/MIME verification,

таким образом это описание применяется также и к упомянутым операциям.

Существует одно принципиальное отличие операций проверки, которые выполняет команда verify, от всех остальных операций проверки: всегда, когда это возможно, делается попытка продолжить работу, в то время как обычно операция проверки прерывается при первой же ошибке. Это позволяет определить все проблемы в цепочке сертификатов.

Операция проверки состоит из ряда отдельных шагов.

Сначала строится цепочка сертификатов от указанного сертификата до корневого сертификата удостоверяющего центра. Если нельзя построить цепочку полностью, это является ошибкой. Цепочка строится с помощью поиска сертификата, на котором подписан рассматриваемый сертификат. Если обнаружен самоподписанный сертификат, он считается корневым сертификатом удостоверяющего центра.

Сам процесс «поиск сертификата, на котором подписан данный сертификат» включает ряд шагов. В версиях OpenSSL до 0.9.5a первый сертификат, поле subject name которого соответствовало полю issuer рассматриваемого сертификата, считался искомым сертификатом. В OpenSSL версии 0.9.6. и позднее все сертификаты, поле subject name которого соответствует полю issuer рассматриваемого сертификата, подвергаются дальнейшему исследованию. Идентификационные компоненты ключа рассматриваемого сертификата (если таковые есть) должны совпадать с соответвующими компонентами искомого сертификата, кроме того, расширение keyUsage искомого сертификата (если таковое есть) должно позволять подписывать сертификаты.

Искомый сертификат прежде всего ищется в списке недоверенных сертификатов, и если он там не найден, дальнейший поиск ведется в списке доверенных сертификатов. Корневой сертификат УЦ всегда ищется в списке доверенных сертификатов; если сертификат, который надо проверить, является корневым сертификатом, то в списке доверенных сертификатов необходимо найти его точную копию.

Второй шаг — проверка расширений каждого недоверенного сертификата на соответствие указанному назначению. Если опция -purpose не указана, такая проверка не проводится. Указанный или «листовой» (leaf) сертификат должен включать расширения, совместимые с указанным назначением, и все остальные сертификаты также должны быть действительными сертификатами удостоверяющих центров. Какие именно расширения требуется, детально описано в разделе 7.6.28.6.

Третий шаг — проверить установки доверия корневому сертификату удостоверяющего центра. Корневой сертификат должен иметь установленное доверие для данного назначения. Для совместимости с предыдущими версиями SSLeay и OpenSSL сертификат без установок доверия считается доверенным для всех назначений.

Последний шаг — проверка сроков действия сертификатов из цепочки. Срок действия сертификатов проверяется исходя из текущего системного времени и дат notBefore и notAfter в сертификате. В это же время проверяются подписи под сертификатами.

Если все шаги выполняются успешно, сертификат считается корректным. Если какой-то из шагов выполнить не удается, сертификат некорректен.

7.6.26.5 Диагностика

Если проверку сертификата выполнить не удается, генерируются сообщения, которые могут озадачить пользователя. Общий вид сообщения об ошибке:

server.pem: /C=AU/ST=Queensland/O=CryptSoft Pty Ltd/CN=Test CA (1024 bit)

error 24 at 1 depth lookup:invalid CA certificate

Первая строка содержит название проверяемого сертификата, за которым следует содержание поля subject name этого сертификата. Вторая строка содержит номер и глубину ошибки. Глубина — это номер сертификата в цепочке, вызвавшего ошибку, причем сам проверяемый сертификат имеет глубину 0, сертификат, на котором подписан проверяемый - глубину 1 и так далее. Заканчивается строка выводом текстовой версии номера ошибки.

Ниже приведен длинный список кодов ошибок и соответствующих сообщений. Список также включает наименование кода, определенное в заголовочном файле `x509_vfy.h`. Некоторые из кодов ошибки определены, но никогда не возникают; они описаны как «неиспользуемые».

Номер ошибки	**Код ошибки**	**Расшифровка**	**Пояснения**
0	X509_V_OK	ok	Операция выполнена успешно
2	X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT	unable to get issuer certificate	Не удается найти сертификат, на котором подписан данный: это происходит, если отсутствует сертификат, на котором подписан данный.
3	X509_V_ERR_UNABLE_TO_GET_CRL	unable to get certificate CRL	Не обнаружен список отзыва сертификатов. Не используется.
5	X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE	unable to decrypt CRL's signature	Подпись под списком отзыва сертификатов не удается расшифровать. Не используется.
6	X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY	unable to decode issuer public key	Открытый ключ в SubjectPublicKeyInfo сертификата не найден.
7	X509_V_ERR_CERT_SIGNATURE_FAILURE	certificate signature failure	Подпись под сертификатом некорректна
8	X509_V_ERR_CRL_SIGNATURE_FAILURE	CRL signature failure	Подпись под списком отзыва сертификатов некорректна. Не используется
9	X509_V_ERR_CERT_NOT_YET_VALID	certificate is not yet valid	Сертификат еще не вступил в действие: дата notBefore еще не наступила.
10	X509_V_ERR_CERT_HAS_EXPIRED	certificate has expired	Срок действия сертификата закончился: дата notAfter уже прошла.
11	X509_V_ERR_CRL_NOT_YET_VALID	CRL is not yet valid	Срок действия списка отзывов сертификатов еще не начался. Не используется
12	X509_V_ERR_CRL_HAS_EXPIRED	CRL has expired	Срок действия списка отзыва сертификатов закончился. Не используется

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Номер ошибки	Код ошибки	Расшифровка	Пояснения
13	X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD	format error in certificate's notBefore field	Поле сертификата notBefore содержит некорректное время
14	X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD	format error in certificate's notAfter field	Поле сертификата notAfter содержит некорректное время
15	X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD	format error in CRL's lastUpdate field	Поле списка отзыва сертификатов lastUpdate содержит некорректное время. Не используется
16	X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD	format error in CRL's nextUpdate field	Поле списка отзыва сертификатов nextUpdate содержит некорректное время. Не используется
17	X509_V_ERR_OUT_OF_MEM	out of memory	Произошла ошибка при распределении памяти. Этого никогда не должно происходить.
18	X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT	self signed certificate	Проверяемый сертификат является самоподписанным и не обнаружен в списке доверенных сертификатов.
19	X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN	self signed certificate in certificate chain	Может быть составлена цепочка из недоверенных сертификатов, но корневой сертификат на данном компьютере не обнаружен.
20	X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY	unable to get local issuer certificate	Сертификат, на котором подписан сертификат, найденный на данном компьютере, не обнаружен. Это обычно означает, что список доверенных сертификатов не полон.
21	X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE	unable to verify the first certificate	Ни одна подпись не может быть проверена, потому что цепочка содержит только один сертификат, не являющийся самоподписанным.
22	X509_V_ERR_CERT_CHAIN_TOO_LONG	certificate chain too long	Длина цепочки сертификатов превосходит указанную максимальную глубину. Неиспользуемая.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Номер ошибки	Код ошибки	Расшифровка	Пояснения
23	X509_V_ERR_CERT_REVOKED	certificate revoked	Сертификат был отозван. Не используется
24	X509_V_ERR_INVALID_CA	invalid CA certificate	Сертификат удостоверяющего центра недействителен. Либо это не сертификат удостоверяющего центра, либо его расширения не соответствуют указанному назначению.
25	X509_V_ERR_PATH_LENGTH_EXCEEDED	path length constraint exceeded	Величина параметра the basicConstraints pathlength превышена.
26	X509_V_ERR_INVALID_PURPOSE	unsupported certificate purpose	Указанный сертификат не может быть использован по указанному назначению.
27	X509_V_ERR_CERT_UNTRUSTED	certificate not trusted	Корневой сертификат удостоверяющего центра не отмечен как доверенный для указанного назначения.
28	X509_V_ERR_CERT_REJECTED	certificate rejected	Корневой сертификат отмечен как отвергнутый для указанного назначения.
29	X509_V_ERR_SUBJECT_ISSUER_MISMATCH	subject issuer mismatch	Сертификат, на котором предположительно был подписан проверяемый сертификат, был отвергнут, потому что значение поля subject name не соответствовало значению поля issuer name проверяемого сертификата. Выводится только в том случае, если указана опция -issuer_checks.

Номер ошибки	Код ошибки	Расшифровка	Пояснения
30	X509_V_ERR_AKID_SKID_MISMATCH	authority and subject key identifier mismatch	Сертификат, на котором предположительно был подписан проверяемый сертификат, был отвергнут, потому что идентификатор subject key присутствует и не соответствует authority key identifier проверяемого сертификата. Выводится только в том случае, если указана опция -issuer_checks.
31	X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH	authority and issuer serial number mismatch	Сертификат, на котором предположительно был подписан проверяемый сертификат, был отвергнут, потому что значения полей issuer name и serial number присутствуют и не соответствуют полю authority key identifier of the current certificate. Выводится только в том случае, если указана опция -issuer_checks.
32	X509_V_ERR_KEYUSAGE_NO_CERTSIGN	key usage does not include certificate signing	Сертификат, на котором предположительно был подписан проверяемый сертификат, был отвергнут, потому что его расширение keyUsage не позволяет подписывать сертификаты.
50	X509_V_ERR_APPLICATION_VERIFICATION	application verification failure	Ошибка, специфичная для приложения. Не используется.

7.6.27 Команда `version`

7.6.27.1 Описание команды

Команда version используется для вывода информации о версии OpenSSL.

7.6.27.2 Формат ввода команды

openssl version [-a] [-v] [-b] [-o] [-f] [-p] [-d]

7.6.27.3 Опции команды

Опция	Описание
-a	Вся информация, то же самое, что выставить все остальные флаги
-v	Текущая версия OpenSSL
-b	Дата выпуска текущей версии OpenSSL
-o	Информация об опциях: различные опции, с которыми собрана библиотека
-f	Флаги компиляции
-p	Платформа, для которой собрана библиотека
-d	Базовый каталог, в который произведена установка библиотеки

7.6.27.4 Примечания

Вывод команды openssl version -a обычно используется в сообщениях об ошибках.

7.6.28 Команда `x509`

7.6.28.1 Описание команды

Команда x509 — многоцелевая утилита для работы с сертификатами. Ее можно использовать для вывода информации о сертификате, преобразования сертификатов в различные формы, подписывания заявок на сертификаты в качестве «мини-УЦ» и редактирования настроек доверия сертификата.

7.6.28.2 Формат ввода команды

openssl x509 [-inform DER|PEM|NET] [-outform DER|PEM|NET] [-keyform DER|PEM] [-CAform DER|PEM] [-CAkeyform DER|PEM] [-in filename] [-out filename] [-serial] [-hash] [-subject_hash] [-issuer_hash] [-ocspid] [-subject] [-issuer] [-nameopt option] [-email] [-ocsp_uri] [-startdate] [-enddate] [-purpose] [-dates] [-checkend num] [-modulus] [-pubkey] [-fingerprint] [-alias] [-noout] [-trustout] [-clrtrust] [-clrreject] [-addtrust arg] [-addreject arg] [-setalias arg] [-days arg] [-set_serial n] [-signkey filename] [-passin arg] [-x509toreq] [-req] [-CA filename] [-CAkey filename] [-CAcreateserial] [-CAserial filename] [-force_pubkey key] [-text] [-certopt option] [-C] [-md_gost12_256|-md_gost12_512|-md_gost94|-md2|-md5|-sha1|-mdc2] [-clrext] [-extfile filename] [-extensions section] [-engine id]

7.6.28.3 Описание опций

Поскольку у этой команды много опций, их описание разбито на несколько разделов.

7.6.28.3.1 Опции ввода, вывода и общего назначения

Опция	**Описание**
-inform DER|PEM|NET	Определяет входной формат. Как правило, команда ожидает сертификат формата X509, но это может измениться, если присутствуют другие опции, например -req. Формат DER — DER-кодировка сертификата, а формат PEM — DER-форма в кодировке base64 с добавленными верхним и нижним ограничителями. Опция NET — непрозрачный формат сервера Netscape, сейчас не рекомендованный к применению
-outform DER|PEM|NET	Определяет выходной формат. Опция имеет те же значения, что и опция -inform.
-in filename	Определяет имя входного файла, из которого следует считать сертификат. Если эта опция не указана, сертификат считывается со стандартного входа.
-out filename	Определяет имя выходного файла. Если эта опция не указана, выходные данные выводятся на стандартный вывод.
-md_gost12_256, -md_gost12_512, -md_gost94, -md2, -md5, -sha1, -mdc2	Использует указанный хэш-алгоритм. Влияет на опции подписи и вывода на экран, которые используют алгоритмы хэширования, такие как -fingerprint, -signkey и –CA. Если опция не указана, то используется алгоритм SHA1.
-engine id	Выбор энджина (по его уникальной строке id) приводит к тому, что x509 делает попытку получить функциональную ссылку на выбранный энджин, таким образом инициализируя его, если это необходимо. После этого энджин устанавливается, как энджин по умолчанию, для всех доступных алгоритмов.

7.6.28.3.2 Опции просмотра сертификатов

Примечание: опции -alias и -purpose также являются опциями просмотра, но описываются в разделе 7.6.28.3.3.

Опция	**Описание**
-text	Выводит сертификат для просмотра в текстовом виде. Полный вывод содержит открытый ключ, алгоритмы подписи, содержание полей subject name и issuer name, серийный номер, все присутствующие расширения и все настройки доверия.
-certopt option	Регулирует формат вывода при использовании опции -text. Аргумент option может быть одной опцией или множеством опций, разделенных запятыми. Для установки значения различных опций свитч -certopt также может использоваться несколько раз. За дополнительной информацией см. раздел 7.6.28.3.6.

Опция	Описание
-noout	Эта опция предотвращает вывод закодированной версии заявки.
-pubkey	Выводит блок SubjectPublicKeyInfo в формате PEM
-modulus	Эта опция выводит значение модуля открытого ключа, содержащегося в сертификате.
-serial	Выводит серийный номер сертификата
-subject_hash	Выводит «хэш» значения поля subject name сертификата. Используется в OpenSSL для создания указателя, позволяющего искать сертификаты в каталоге по значению поля subject name.
-issuer_hash	Выводит «хэш» значения поля issuer name сертификата.
-ocspid	Выводит OCSP-идентификатор (хэш имени субъекта и открытого ключа).
-hash	Синоним опции -hash для обеспечения обратной совместимости.
-subject_hash_old	Выводит “хэш” имени субъекта сертификата, используя старый алгоритм, который применялся в OpenSSL до версии 1.0.0.
-issuer_hash_old	Выводит “хэш” имени издателя сертификата, используя старый алгоритм, который применялся в OpenSSL до версии 1.0.0.
-subject	Выводит значение поля subject name.
-issuer	Выводит значение поля issuer name.
-nameopt option	Опция указывает, как должны выводиться значения полей subject и issuer. Аргументом может быть одна опция или несколько, разделенных запятыми. Для установки нескольких опций можно также несколько раз использовать свитч -nameopt. Для получения дополнительной информации см. раздел 7.6.28.3.5.
-email	Выводит адрес(а) электронной почты, если таковые указаны.
-ocsp_uri	Выводит URI, на который следует отправлять OCSP-запросы по проверке статуса данного сертификата
-startdate	Выводит дату начала срока действия сертификата notBefore.
-enddate	Выводит дату окончания срока действия сертификата notAfter.
-dates	Выводит даты начала и окончания срока действия сертификата.
-checkend arg	Проверяет, закончится ли срок действия сертификата в течение количества секунд, определенного аргументом arg. Если да, выводит 1, если нет, выводит 0
-fingerprint	Выводит хэш-сумму DER-закодированной версии всего сертификата (см. раздел 7.6.28.5).
-C	Эта опция выводит сертификат в виде файла на языке C.

7.6.28.3.3 Настройки доверия

Пожалуйста, учтите, что эти опции в настоящее время являются экспериментальными и могут измениться.

Доверенный сертификат — это обычный сертификат, к которому добавлена некоторая дополнительная информация, такая как разрешенные или запрещенные назначения сертификата и его «алиас».

Обычно при подтверждении сертификата хотя бы один сертификат в цепочке должен быть «доверенным». По умолчанию доверенный сертификат должен храниться на локальном ком-

пьютере и представлять собой корневой сертификат удостоверяющего центра. Любая цепочка сертификатов, заканчивающаяся этим сертификатом, при таких условиях годится для любых целей.

В настоящее время настройки доверия используются только для корневых сертификатов удостоверяющих центров. Они предоставляют более гибкий контроль над назначениями корневого сертификата удостоверяющего центра. Например, такой сертификат может быть доверенным для использования SSL-клиентом, но не SSL-сервером.

См. раздел 7.6.26 для получения дополнительной информации о значении настроек доверия.

Будущие версии OpenSSL будут распознавать настройки доверия любых сертификатов, не только сертификатов удостоверяющего центра.

Опция	**Описание**
-trustout	Эта опция заставляет утилиту x509 создать доверенный сертификат в качестве выходных данных. В качестве входных данных может быть как обычный сертификат, так и доверенный, но по умолчанию выходными данными является обычный сертификат, и все настройки доверия отбрасываются. С использованием этой опции создается доверенный сертификат. Доверенный сертификат создается автоматически, если модифицируются какие-либо настройки доверия.
-setalias arg	Устанавливает алиас сертификата. Это позволяет ссылаться на сертификат по алиасу, например «сертификат Иванова».
-alias	Выводит алиас сертификата, если таковой существует.
-clrtrust	Очищает все дозволенные или доверенные назначения сертификата.
-clrreject	Очищает все запрещенные или отвергнутые назначения сертификата.
-addtrust arg	Добавляет доверенное назначение сертификата. Здесь может быть использовано любое наименование объекта, но в настоящее время используется только clientAuth (использование с SSL-клиентом), serverAuth (использование с SSL-сервером) и emailProtection (электронная почта формата S/MIME). Другие OpenSSL-приложения могут определять дополнительные назначения.
-addreject arg	Добавляет запрещенное назначение сертификата. Опция принимает те же значения, что и опция -addtrust.
-purpose	Эта опция выполняет тестирование расширений сертификатов и выводит результаты. Для получения дополнительной информации см. раздел 7.6.28.6.

7.6.28.3.4 Опции подписания сертификатов

Команда x509 может использоваться для подписания сертификатов и заявок; таким образом, она может вести себя как «мини-УЦ».

Опция	Описание
-signkey filename	Эта опция создает самоподписанный файл при помощи указанного закрытого ключа. Если входной файл является сертификатом, опция устанавливает значение поля issuer name равным значению поля subject name (т.е. делает его самоподписанным), заменяет открытый ключ на указанный и меняет даты начала и конца срока действия. Дата начала действия устанавливается в текущее время, а дата конца действия устанавливается во время, определенное опцией -days. Все расширения сертификата сохраняются, если не указана опция -clrext. Если входной файл является заявкой на сертификат, то создается самоподписанный сертификат, использующий указанный закрытый ключ и значение поля subject name из заявки.
-passin arg	Указывает, где содержится пароль для ключа. Для получения дополнительной информации по формату аргумента arg см. раздел 7.4.
-clrext	Удаляет все расширения из сертификата. Эта опция используется, когда сертификат создается из другого сертификата (например с помощью опции -signkey или -CA). Если эта опция не указана, все расширения сохраняются.
-keyform PEM|DER	Указывает формат (DER или PEM) файла закрытого ключа, используемого в опции -signkey.
-days arg	Указывает срок действия сертификата. По умолчанию 30 дней.
-x509toreq	превращает сертификат в заявку. Опция -signkey используется для передачи требуемого закрытого ключа.
-req	По умолчанию в качестве входных данных ожидается сертификат. Если указана данная опция, вместо сертификата ожидается заявка на сертификат.
-set_serial n	Указывает используемый серийный номер. Эта опция может использоваться с опциями -signkey или -CA. Если используется вместе с опцией -CA, то файл серийного номера (определенный опциями -CAserial или -CAcreateserial) не используется. Серийный номер может быть десятичным или шестнадцатиричным (с префиксом 0x). Указывать отрицательные серийные номера можно, но не рекомендуется.
-CA filename	Указывает сертификат удостоверяющего центра, который следует использовать для подписывания. Когда указывается эта опция, команда x509 работает как «мини-УЦ». При указании этой опции входной файл подписывается на указанном сертификате удостоверяющего центра, то есть поле issuer name входного файла устанавливается в значение поля subject name указанного сертификата УЦ, формируется цифровая подпись под входным файлом с использованием закрытого ключа, соответствующего указанному сертификату УЦ. Эта опция, как правило, указывается вместе с опцией -req. Без указания опции -req входным файлом является сертификат, который необходимо сделать самоподписанным.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
-CAkey filename	Указывает закрытый ключ, соответствующий сертификату УЦ, на котором следует подписывать входной сертификат. Если эта опция не указана, считается, что закрытый ключ включен в файл сертификата УЦ.
-CAserial filename	Устанавливает, какой файл серийного номера УЦ следует использовать. Если указывается опция -CA при подписывании сертификата, она использует серийный номер, указанный в файле. Этот файл состоит из одной строки, содержащей четное количество шестнадцатиричных цифр. После каждого применения этой опции серийный номер увеличивается на единицу и снова записывается в этот файл. Умолчательное значение имени файла состоит из имени файла сертификата УЦ (взятого без расширения) с расширением .srl. Например, если файл сертификата УЦ называется mycacert.pem, ожидается существующий файл серийного номера mycacert.srl.
-CAcreateserial	При указании этой опции, если файл серийного номера УЦ не существует, он создается. Файл будет содержать серийный номер 02, а подписанный сертификат получит серийный номер 1. Как правило, если указана опция -CA и файла серийного номера не существует, это является ошибкой.
-extfile filename	Файл, содержащий расширения сертификатов, которые следует использовать. Если не указан, к сертификату не добавляется никаких расширений.
-extensions section	раздел конфигурационного файла, из которого следует добавлять расширения в сертификаты. Если эта опция не указана, расширения должны либо содержаться в непоименованном (умолчательном) разделе, либо умолчательный раздел должен содержать переменную ”extensions”, содержащую наименование соответствующего раздела.
-force_pubkey key	При создании сертификата поместить в него открытый ключ key вместо ключа, взятого из сертификата или запроса на сертификат. Эта опция полезна при создании сертификатов, когда алгоритм не может подписывать запросы, например, DH. Формат key может быть указан с помощью опции –keyform.

7.6.28.3.5 Опции именования

Командно-строчный свитч nameopt определяет, как выводятся поля subject name и issuer name. Если ни одного свитча nameopt не указано, используется умолчательный «однострочный» формат, совместимый с предыдущими версиями OpenSSL. Каждая опция подробно описана внизу, перед каждой опцией может стоять «-» для ее отключения. Как правило, используются только первые четыре.

Опция	Описание
compat	Использовать старый формат. Это эквивалентно отсутствию указания опций именования.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
RFC2253	Выводит имена, совместимые с определенным в RFC2253 эквивалентом опций esc_2253, esc_ctrl, esc_msb, utf8, dump_nostr, dump_unknown, dump_der, sep_comma_plus, dn_rev и sname.
oneline	Однострочный формат, более читабельный, чем RFC2253. Эквивалентно определению опций esc_2253, esc_ctrl, esc_msb, utf8, dump_nostr, dump_der, use_quote, sep_comma_plus_space, space_eq и sname.
multiline	Многострочный формат. Эквивалентен определению опций esc_ctrl, esc_msb, sep_multiline, space_eq, lname и align.
esc_2253	Экранировать «специальные» символы, требуемые RFC2253 в поле, то есть ,+«»;. Кроме того, # экранируется в начале строки, а пробел — в начале или в конце строки.
esc_ctrl	Экранировать контрольные символы, т.е. символы с ASCII-значениями, меньшими 0x20 (пробел), и символ удаления (0x7f). Они экранируются с использованием определенной в RFC2253 нотации \XX notation (где XX — две шестнадцатеричных цифры, представляющие значение символа).
esc_msb	Экранировать символы с ненулевым старшим (most significant) битом, то есть символы, ASCII-значения которых превосходят 127. Эту опцию не рекомендуется использовать при работе с сертификатами, содержащими кириллицу
use_quote	Экранирует некоторые символы, окружая всю строку символов символами без указания этой опции все экранирование выполняется с помощью символа \.
utf8	Сначала перевести все строки в UTF-формат. Это требуется в RFC2253. Если вам повезло и у вас есть UTF-8-совместимый терминал, то использование этой опции (без установки esc_msb) может привести к корректному выводу многобайтных (международных) символов. Если эта опция не присутствует, многобайтные символы, превосходящие 0xff, будут представлены в формате \UXXXX для шестнадцатибитных и \WXXXXXXXX для тридцатидвухбитных. Кроме того, если эта опция отключена, любые UTF-8-строки сначала будут переведены в свою символьную форму.
ignore_type	Эта опция вообще не пытается интерпретировать многобайтные символы. Их содержательные октеты просто дампируются так, как если бы один октет содержал один символ. Это полезно для диагностических целей, но приведет к весьма странно выглядящим выходным данным.
show_type	Показывает тип символьной строки ASN.1. Тип указывается до содержания поля. Например ”BMPSTRING: Hello World”.
dump_der	Когда эта опция установлена, любые поля, которые нуждаются в шестнадцатиричном дампировании, будут дампированы с использованием DER-кодировки. Если опция не установлена, будут выведены только октеты содержания. Обе опции используют описанный в RFC2253 формат #XXXX...

Опция	Описание
dump_nostr	Дампировать типы несимвольных строк (например OCTET STRING). Если эта опция не указана, типы несимвольных строк будут выводиться так, как будто каждый октет содержания представляет один символ.
dump_all	Дампировать все поля. Эта опция при использовании с dump_der позволяет однозначно определить DER-кодирование структуры.
dump_unknown	Дампировать все поля, чьи OID не распознаются OpenSSL.
sep_comma_plus, sep_comma_plus_space, sep_semi_plus_space, sep_multiline	Эти опции определяют разделители полей. Первый символ - разделитель для RDN и второй — для многократных AVA (многократные AVA встречаются очень редко, и их использование не рекомендуется). Опции, заканчивающиеся на слово space, дополнительно помещают пробел после разделителя для лучшей читабельности. Опция sep_multiline использует символ LF для разделителя RDN и окруженный пробелами + в качестве разделителя AVA. Кроме того, она обусловливает помещение в начале каждого поля отступа в четыре символа.
dn_rev	Изменить порядок полей в DN на противоположный. Это требование RFC2253. В качестве дополнительного эффекта эта опция также обращает порядок многократных AVA, но это позволительно.
nofname, sname, lname, oid	Эти опции влияют на формат вывода названия поля. Опция nofname вообще не выводит соответствующее поле. Опция sname использует форму «короткого имени» (например CN вместо commonName). Опция lname использует полное наименование. Опция oid представляет OID в численной форме и полезна для диагностических целей.
align	Выравнивает значения полей для более читабельного вывода. Может использоваться только вместе с опцией sep_multiline.
space_eq	Окружает пробелами символ «=», следующий за именем поля.

7.6.28.3.6 Текстовые опции

Можно регулировать не только формат вывода имен, но и набор выводимых полей, используя опции certopt, если присутствует опция text. По умолчанию выводятся все поля.

Опция	Описание
compatible	Использовать старый формат. Это эквивалентно отсутствию указания опций вывода.
no_header	Не выводить информацию о заголовках, т.е. о строках Certificate и Data.
no_version	Не выводить номер версии
no_serial	Не выводить серийный номер
no_signame	Не выводить используемый алгоритм подписи
no_validity	Не выводить срок действия, т.е. поля notBefore и notAfter.
no_subject	Не выводить поле subject name.
no_issuer	Не выводить поле issuer name.
no_pubkey	Не выводить открытый ключ.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Опция	Описание
no_sigdump	Не выводить шестнадцатеричный дамп подписи под сертификатом.
no_aux	Не выводить настройки доверия сертификата.
no_extensions	Не выводить расширения версии 3 формата X509.
ext_default	Сохранить умолчательное поведение расширений; попытаться вывести неподдерживаемые расширения сертификатов.
ext_error	Вывести сообщение об ошибке для неподдерживаемых расширений сертификатов.
ext_parse	Вывести неподдерживаемые расширения в виде ASN.1-структуры.
ext_dump	Вывести шестнадцатеричный дамп неподдерживаемых расширений.
ca_default	Значение, используемое командой ca, эквивалентно набору опций no_issuer, no_pubkey, no_header, no_version, no_sigdump и no_signame.

7.6.28.4 Примеры

Вывести содержание сертификата на экран:

```
openssl x509 -in cert.pem -noout -text
```

Вывести серийный номер сертификата:

```
openssl x509 -in cert.pem -noout -serial
```

Вывести поле subject name сертификата:

```
openssl x509 -in cert.pem -noout -subject
```

Вывести поле subject name сертификата в RFC2253-формате:

```
openssl x509 -in cert.pem -noout -subject -nameopt RFC2253
```

Вывести поле subject name сертификата в однострочном формате на терминале, поддерживающем UTF-8:

```
openssl x509 -in cert.pem -noout -subject -nameopt oneline,-esc_msb
```

Вывести MD5-отпечаток сертификата:

```
openssl x509 -in cert.pem -noout -fingerprint
```

Вывести SHA1-отпечаток сертификата:

```
openssl x509 -sha1 -in cert.pem -noout -fingerprint
```

Перевести сертификат из PEM-формата в DER-формат:

```
openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
```

Получить из сертификата заявку:

```
openssl x509 -x509toreq -in cert.pem -out req.pem -signkey key.pem
```

Превратить заявку на сертификат в самоподписанный сертификат, используя расширения для сертификата УЦ:

```
openssl x509 -req -in careq.pem -extfile openssl.cnf -extensions
v3_ca -signkey key.pem -out cacert.pem
```

Подписать заявку на сертификат, используя сертификат УЦ из предыдущего примера, и добавить расширения пользовательского сертификата:

```
openssl x509 -req -in req.pem -extfile openssl.cnf -extensions
v3_usr -CA cacert.pem -CAkey key.pem -CAcreateserial
```

Установить для сертификата доверие для использования с SSL-клиентом и установить для него алиас Steve's Class 1 CA:

```
openssl x509 -in cert.pem -addtrust clientAuth -setalias "Steve's
Class 1 CAout trust.pem
```

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

7.6.28.5 Примечания

PEM-формат использует следующий вид ограничителей:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

Кроме того, обрабатываются файлы, содержащие следующий вид ограничителей:

```
-----BEGIN X509 CERTIFICATE-----
-----END X509 CERTIFICATE-----
```

Вид ограничителей для доверенных сертификатов:

```
-----BEGIN TRUSTED CERTIFICATE-----
-----END TRUSTED CERTIFICATE-----
```

Перевод в формат UTF-8, используемый с опциями именования, предполагает, что в переменных типа T61Strings используется кодировка ISO8859-1. Это не так, но так работают Netscape и Microsoft IE, а также множество сертификатов. Поэтому хотя это предположение и некорректно, его использование позволяет корректно вывести большинство сертификатов.

Опция -fingerprint выводит хэш-сумму DER-закодированного сертификата. Эта хэш-сумма обычно называется «отпечатком пальца». Вследствие природы хэш-сумм «отпечаток пальца» уникален для каждого сертификата, и два сертификата с одним и там же «отпечатком» могут считаться одним и тем же сертификатом.

Netscape использует алгоритм хэширования MD5, а Microsoft IE использует SHA1.

Опция -email просматривает поле subject name и расширение subject alternative name. Выводятся только уникальные почтовые адреса: опция не выводит один и тот же почтовый адрес дважды.

7.6.28.6 Расширения сертификатов

Опция -purpose проверяет расширения сертификатов и определяет, для чего сертификат может быть использован. Выполняемые проверки довольно сложны и включают различные способы обращения с некорректными сертификатами и программами.

Тот же код используется при проверке недоверенных сертификатов в цепочках, поэтому этот раздел полезен в том случае, если цепочка признана некорректной при использовании команды -verify.

Флаг расширения basicConstraints сертификата удостоверяющего центра используется для определения того, может ли сертификат использоваться как сертификат удостоверяющего центра. Если установлен флаг сертификата удостоверяющего центра true, то это сертификат УЦ, если этот флаг false, то это не сертификат УЦ. Все сертификаты УЦ должны иметь значение этого флага true.

Если сертификат является сертификатом версии V1 (то есть не имеет расширений) и он самоподписан, он считается сертификатом УЦ, но выводится предупреждение об этом. Это способ обхода проблемы с корневыми сертификатами Verisign, которые являются самоподписанными сертификатами версии V1.

Если присутствует расширение keyUsage, на возможные назначения сертификата накладываются дополнительные ограничения. Сертификат УЦ должен иметь установленные бит keyCertSign, если в нем присутствует данное расширение.

Расширение extended key usage накладывает дальнейшие ограничения на возможные назначения сертификатов. Если это расширение присутствует (критичное или нет), ключ может использоваться только для указанных назначений.

Ниже приводится полное описание каждой проверки. Приведенные выше комментарии о basicConstraints, keyUsage и сертификатов версии 1 применимы ко всем сертификатам удостоверяющих центров.

Назначение	**Требования**
SSL Client	Расширение extended key usage должно отсутствовать или включать OID web client authentication. Расширение keyUsage должно отсутствовать или иметь установленный бит digitalSignature. Тип сертификата Netscape должен отсутствовать или иметь установленный бит SSL client.
SSL Client CA	Расширение extended key usage должно отсутствовать или включать OID web client authentication. Тип сертификата Netscape должен отсутствовать или иметь установленный бит SSL CA, это используется для обхода ситуации, если отсутствует расширение basicConstraints.
SSL Server	Расширение extended key usage должно отсутствовать или включать OID web server authentication и/или один из SGC OID. Расширение keyUsage должно отсутствовать или иметь установленный бит digitalSignature или keyEncipherment (или оба). Тип сертификата Netscape должен отсутствовать или иметь установленный бит SSL server.
SSL Server CA	Расширение extended key usage должно отсутствовать или включать OID web server authentication и/или один из SGC OID. Тип сертификата Netscape должен отсутствовать или иметь установленный бит SSL CA, это используется для обхода ситуации, если отсутствует расширение basicConstraints.
Netscape SSL Server	Чтобы Netscape SSL-клиент мог соединиться с SSL-сервером, сертификат должен иметь установленный бит keyEncipherment если присутствует расширение keyUsage. Это не всегда корректно, потому что некоторые шифр-сьюты используют этот ключ для создания цифровой подписи. Все остальное так же, как для обычного SSL-сервера.
Common S/MIME Client Tests	Расширение extended key usage должно отсутствовать или включать OID email protection. Тип сертификата Netscape должен отсутствовать или иметь установленый бит S/MIME. Если бит S/MIME не установлен в типе сертификата Netscape, в качестве альтернативы допустим установленный бит SSL client, но выводится предупреждение; это связано с тем, что некоторые сертификаты Verisign не устанавливают бит S/MIME.
S/MIME Signing	В дополнение к обычным проверкам для S/MIME-клиента должен быть установлен бит digitalSignature, если присутствует расширение keyUsage.
S/MIME Encryption	В дополнение к обычным проверкам для S/MIME-клиента должен быть установлен бит keyEncipherment, если присутствует расширение keyUsage.
S/MIME CA	Расширение extended key usage должно отсутствовать или включать OID email protection. Тип сертификата Netscape должен отсутствовать или иметь установленый бит S/MIME CA, это используется для обхода ситуации, если отсутствует расширение basicConstraints.

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения

Назначение	Требования
CRL Signing	Расширение keyUsage должно отсутствовать или иметь установленный бит CRL signing.
CRL Signing CA	Проводятся обычные проверки сертификата удостоверяющего центра. Но в этом случае должно присутствовать расширение basicConstraints.

7.7 Команды хэширования

Опция	Описание
md2	MD2 алгоритм хэширования
md5	MD5 алгоритм хэширования
mdc2	MDC2 алгоритм хэширования
rmd160	RMD-160 алгоритм хэширования
sha	SHA алгоритм хэширования
sha1	SHA-1 алгоритм хэширования
sha224	SHA-224 алгоритм хэширования
sha256	SHA-256 алгоритм хэширования
sha384	SHA-384 алгоритм хэширования
sha512	SHA-512 алгоритм хэширования

7.8 Команды кодирования и шифрования

Опция	Описание
base64	Кодирование Base64
bf bf-cbc bf-cfb bf-ecb bf-ofb	Шифр Blowfish
cast cast-cbc	Шифр CAST
cast5-cbc cast5-cfb cast5-ecb cast5-ofb	Шифр CAST5
des des-cbc des-cfb des-ecb des-ede des-ede-cbc des-ede-cfb des-ede-ofb des-ofb	Шифр DES
des3 desx des-ede3 des-ede3-cbc des-ede3-cfb des-ede3-ofb	Шифр тройной DES
idea idea-cbc idea-cfb idea-ecb idea-ofb	Шифр IDEA
rc2 rc2-cbc rc2-cfb rc2-ecb rc2-ofb	Шифр RC2
rc4	Шифр RC4
rc5 rc5-cbc rc5-cfb rc5-ecb rc5-ofb	Шифр RC5

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий сопроводительного докум. и дата	Подпись	Дата
Изм.	измененных	замененных	новых	аннулированных					

Порядковый № изменения	Подпись лица, ответственного за изменение	Дата внесения изменения