

TRUSTED TLS 3.0 Standard

РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ

Оглавление

Подключение engine-модуля поддержки СКЗИ КриптоПро CSP	3
Перечень команд утилиты openssl	4
Определение версии OpenSSL.....	4
Определение списка команд	5
Просмотр доступных SSL/TLS-шифров.....	5
Генерация закрытого ключа	6
Генерация запроса на сертификат	7
Операции с сертификатами.....	8
Работа с файлами PKCS#7 и CMS форматов	8
Подпись файла.....	8
Добавление подписи	8
Проверка подписи	9
Извлечение подписанных данных	9
Шифрование файла	9
Расшифровывание файла.....	10
Работа с хэшами	10
Вычисление хэша	10
Подпись хэша	10
Проверка подписи хэша.....	11
Шифрование данных	11
Измерение производительности	11
SSL/TLS клиент.....	12
SSL/TLS сервер	12
Перечень команд утилиты ctgostcp_util	14
Техническая поддержка	16

Подключение engine-модуля поддержки СКЗИ КриптоПро CSP

Для выполнения в утилите **openssl** криптографических операций через СКЗИ КриптоПро CSP необходимо подключить engine-библиотеку ctgostcp. Для этого в конфигурационном файле **openssl.cnf** необходимо указать следующее:

1. В глобальном контексте, перед первой секцией, разместить директиву

```
openssl_conf=openssl_def
```

2. Секция **[openssl_def]**, включающая глобальные установки по умолчанию, должна содержать директиву

```
engines = engines_section
```

3. Секция **[engines_section]**, включающая описание загружаемых engine-модулей, должна содержать директиву

```
ctgostcp = engine_section
```

4. Секция **[engine_section]** включает параметры engine`а. В качестве параметров могут выступать, например: `engine_id`, `init`, `dynamic_path`, `module_path`, `default_algorithms`.

Ниже приводится пример необходимых директив для конфигурирования engine-модуля ctgostcp:

```
openssl_conf=openssl_def
...
[openssl_def]
engines = engines_section

[engines_section]
ctgostcp = ctgostcp_section

[ctgostcp_section]
engine_id = ctgostcp
dynamic_path = ctgostcp
default_algorithms = ALL
```

В дистрибутиве Trusted TLS 3 в каталоге `/opt/TrustedTLS-3/bin/ctsamples` расположен пример такого файла **ctgostcp.cnf**.

Перед вызовом утилиты **openssl**, если необходимо изменить умолчания, можно указать расположение конфигурационного файла, указав его в переменной окружения **OPENSSL_CONF**.

Так как в примере в директиве `dynamic_path` указано не полное имя engine-файла, необходимо также определить переменную окружения **OPENSSL_ENGINES**, указывающую на каталог его расположения, например,

для Linux:

```
OPENSSL_ENGINES=/opt/TrustedTLS-3/lib/engines
```

```
export OPENSSL_ENGINES
```

для Windows:

```
set OPENSSL_ENGINES=C:\opt\TrustedTLS-3\bin
```

Перечень команд утилиты openssl

Данное руководство основано на официальной документации по [списку](#) команд утилиты openssl, не является полным и в большей степени отражает специфику вызова команд в применении к engine-библиотеке ctgostcp.

Также ознакомиться с примерами использования OpenSSL можно на ресурсе xgu.ru.

Определение версии OpenSSL

Определить версию можно командой

```
>openssl version
```

Команда выдает информацию, например

```
OpenSSL 1.0.1c 10 May 2012
```

Расширенная информация получается по команде

```
>openssl version -a
```

Команда выдает информацию, например

```
OpenSSL 1.0.1c 10 May 2012
built on: Thu Oct 11 09:18:51 2012
platform: VC-WIN32
options: bn(64,32) rc4(8x,mmx) des(idx,cisc,2,long) idea(int) blowfish(idx)
compiler: cl /MD /Ox /O2 /Ob2 -DOPENSSL_THREADS -D_DSO_WIN32 -W3 -Gs0 -
GF -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -
DL_ENDIAN -DUNICODE -D_UNICODE -D_CRT_SECURE_NO_DEPRECATED -
DDIGTLICENSE_STATIC -DOPENSSL_BN_ASM_PART_WORDS -
DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -
DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -
```

```
DMD5_ASM      -DRMD160_ASM      -DAES_ASM      -DVPAES_ASM      -  
DWHIRLPOOL_ASM -DGHASH_ASM      -DOPENSSL_USE_APPLINK -I.      -  
DOPENSSL_NO_RC5  -DOPENSSL_NO_MD2  -DOPENSSL_NO_KRB5  -  
DOPENSSL_NO_JPAKE -DOPENSSL_NO_STATIC_ENGINE  
OPENSSLDIR: "/usr/local/ssl"
```

Определение списка команд

Для того, чтобы вывести список команд, поддерживаемых утилитой **openssl**, запустите его со следующими параметрами:

```
>openssl list-standard-commands  
>openssl list-cipher-commands  
>openssl list-message-digest-commands
```

Для вывода списка поддерживаемых алгоритмов используются следующие команды:

```
>openssl list-cipher-algorithms  
>openssl list-message-digest-algorithms  
>openssl list-public-key-algorithms
```

В утилите **openssl** не предусмотрено опции для вывода списка поддерживаемых опций команды, но если запустить утилиту, указав неверную опцию команды, то выведется список поддерживаемых опций команды, например,

```
>openssl cms -help  
  
unknown option '-help'  
options are  
Usage cms [options] cert.pem ...  
where options are  
-encrypt      encrypt message  
-decrypt      decrypt encrypted message  
-sign         sign message  
-verify       verify signed message  
-cmsout       output CMS structure  
...  
...  
-engine e     use engine e, possibly a hardware device.  
-passin arg   input file pass phrase source  
-rand file;file;...  
               load the file (or the files in the directory) into  
               the random number generator  
cert.pem     recipient certificate(s) for encryption
```

Просмотр доступных SSL/TLS-шифров

Для просмотра доступных SSL/TLS-шифров нужно использовать команду **ciphers**. Далее приводятся примеры использования данной команды.

Получить все доступные SSL/TLS-шифры:

```
>openssl ciphers -v ALL
```

Получить шифры TLS версии 1.0 и SSL 3.0 (вывести только шифры TLS 1.0 или только SSL 3.0 нельзя, т.к. openssl не отделяет их друг от друга):

```
>openssl ciphers -v -tls1
```

Получить шифры длиной больше 128 битов (high ciphers):

```
>openssl ciphers -v "HIGH"
```

Получить шифры длиной больше 128 битов, использующие AES:

```
>openssl ciphers -v "AES+HIGH"
```

Генерация закрытого ключа

Закрытый ключ с размещением его в хранилище КриптоПро CSP можно создать командой

```
>openssl genpkey \  
-engine ctgostcp \  
-algorithm ALGORITHM \  
-pkeyopt "container:CONTAINER" \  
-pkeyopt "algid:ALGID" \  
-out KEY_FILE
```

Где

- **ALGORITHM** – схема формируемого ключа; возможные значения: gost2001, gost2012-256 и gost2012-512.
- **CONTAINER** – наименование ключевого контейнера в [формате](#) КриптоПро CSP.
- **ALGID** – тип ключа; возможные значения см. ниже.
- **KEY_FILE** - имя выходного псевдоключевого файла.

В полный перечень параметров и их значений, поддерживаемых в -pkeyopt, входят:

- "keyid" и "container" – название ключевого контейнера в формате КриптоПро CSP;
- "password" и "pin" – пароль для доступа к ключевому контейнеру;
- "savepass" – требуется ли записывать пароль в key-файл, возможные значения:
 - "yes",
 - "no" (по умолчанию);
- "keyset" - расположение ключа, возможные значения:
 - "user" (по умолчанию) – пользователя,
 - "machine" – компьютера;
 - "auto" и "all" – поиск ключа в обоих хранилищах (по умолчанию при поиске);
- "algid" – тип ключа, возможные значения:
 - "exch" – подписи и шифрования (по умолчанию при создании),
 - "sign" – только подписи.
 - "auto" – поиск ключа (по умолчанию при поиске)

- "exportable" – должен ли создаваемый ключ быть экспортируемым, возможные значения:
 - "yes"
 - "no"
- "existing" – использовать ли существующий ключ, возможные значения:
 - "yes"
 - "no" (по умолчанию)
- "silent" – запрет интерактивного взаимодействия, возможные значения:
 - "yes"
 - "no" (по умолчанию)

Если в команде присутствует параметр `existing` со значением «yes», то создания нового ключевого контейнера КриптоПро CSP не происходит, - псевдоключевой файл будет создан для существующего закрытого ключа.

Часть этих параметров можно использовать в командах, использующих параметр - **keyform ENGINE** в связке с опцией, указывающей на закрытый ключ, например, допустима связка

-keyform ENGINE -engine ctgostcp -inkey "pin:password,file:filename.key"

Пример генерации ключей приведен в поставляемом с дистрибутивом скрипте **create-key** в каталоге `/opt/TrustedTLS-3/bin/ctsamples`.

Для удаления ключей предлагается использовать утилиту КриптоПро CSP **csptest**, например,

> **csptest -keyset -deletekeyset -container "CONTAINER"**

Генерация запроса на сертификат

Некоторые примеры, приводимые ниже, представлены в скрипте **create-key** в каталоге `/opt/TrustedTLS-3/bin/ctsamples`.

Для генерации файла **req_file** запроса на сертификат с использованием файла **key_file** закрытого ключа и имени субъекта **SUBJ** необходимо выполнить команду

> **openssl req -new -engine ctgostcp -subj "SUBJ" -key key_file -out req_file**

При выполнении этой команды при необходимости будет запрошен пароль доступа **key_pin** к закрытому ключу.

Добавление опции `-x509` позволит вместо запроса на сертификат создать на выходе самоподписанный сертификат.

Проверка запроса на сертификат и вывод заявки в текстовом виде могут быть выполнены по команде

> **openssl req -verify -noout -text -in req_file**

Операции с сертификатами

Команда **x509** позволяет производить большое количество операций над сертификатами.

Для создания самоподписанного сертификата **cert_file** на базе запроса на него **req_file** и закрытого ключа подписчика **key_file** можно использовать команду

```
>openssl x509 -engine ctgostcp -req -in req_file -out cert_file -signkey key_file
```

Для создания сертификата удостоверяющим центром вместо опции **-signkey** необходимо использовать опции **-CA** и **-CAkey**.

Работа с файлами PKCS#7 и CMS форматов

Основные примеры, приводимые ниже, представлены в скрипте **crypto-tests**, который находится в дистрибутиве Trusted TLS в каталоге **/opt/TrustedTLS-3/bin/ctsamples**.

Подпись файла

Чтобы получить подпись файла **data_file** в виде отдельной подписи в кодировке BASE64 в файле **data_file.sig**, с использованием файла **cert_file** сертификата подписчика и с указанием на файл **key_file** закрытого ключа, нужно выполнить команду

```
>openssl cms -sign -engine ctgostcp -in data_file -binary -signer cert_file \
    -inkey key_file -out data_file.sig -outform PEM
```

В случае создания совмещенной подписи требуется указать опцию **-nodetach**.

Альтернативно можно использовать команду

```
>openssl cms -sign -engine ctgostcp -in data_file -binary -signer keycert_file \
    -out data_file.sig -outform PEM
```

В этом случае в файле **keycert_file** должен находиться закрытый ключ и сертификат. Передача пароля доступа через опцию **-passin** не поддерживается.

При необходимости запрос пароля доступа к закрытому ключу будет произведен штатными средствами (например, через интерфейс СКЗИ КриптоПро CSP).

Если определена переменная окружения **OPENSSL_ENGINES** или **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine`-ов, то указание опции **-engine** не требуется.

Добавление подписи

Добавление подписи к уже существующей осуществляется по команде:

```
>openssl cms -resign -engine ctgostcp -nocerts -content data_file \
    -in data_file.sig -inform PEM \
    -signer cert_file -inkey key_file \
    -out data_file.sig -outform PEM
```


Если наложение второй подписи происходит тем же подписчиком, то, чтобы не происходило ошибки выполнения при добавлении уже существующего сертификата, необходимо указать опцию **-nocerts**.

Если добавление производится к совмещенной подписи, то использование опции **-content** не требуется.

Проверка подписи

Для проверки отделенной от файла **data_file** подписи **data_file.sig** в CMS-формате необходимо выполнить команду:

```
>openssl cms -verify -engine ctgostcp -content data_file -in data_file.sig -inform PEM -CAfile cert_file
```

При проверке совмещенной подписи использование опции **-content** не требуется.

Если определена переменная окружения **OPENSSL_ENGINES** или **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine`-ов, то указание опции **-engine** не требуется.

Извлечение подписанных данных

Для извлечения сертификатов и подписанных данных из файла **data_file.sig** совмещенной подписи в CMS-формате необходимо выполнить команду:

```
>openssl cms -verify -engine ctgostcp -no_signer_cert_verify -in data_file.sig -inform PEM \
-out detached.data_file -certsout detached.certs.data_file.pem
```

В файле **detached.data_file** будет содержимое подписанных данных, а в файле **detached.certs.data_file.pem** - сертификаты, использованные при их подписании. Если не важен результат полноценной проверки подписи, то можно указать опцию **-no_signer_cert_verify**, которая позволит сократить время извлечения данных за счет отказа от проверки сертификатов подписчиков.

Если определена переменная окружения **OPENSSL_ENGINES** или **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine`-ов, то указание опции **-engine** не требуется.

Шифрование файла

Шифрование файла **data_file** в адрес сертификатов получателей **cert1_file** и **cert2_file**, с сохранением результата в файле **data_file.enc** в кодировке BASE64 нужно выполнить команду:

```
>openssl cms -encrypt -engine ctgostcp -gost89 -in data_file -binary -stream \
-out data_file.enc -outform PEM cert1_file cert2_file
```

Расшифровывание файла

Извлечение исходных данных, зашифрованных перед передачей адресату и сохраненных в файле **data_file.enc**, на сертификате получателя **cert_file**, с указанием на файл закрытого ключа **key_file** и пароля доступа к его хранилищу **key_pin** и сохранением результата в файл **decrypt.data_file** нужно выполнить команду:

```
>openssl cms -decrypt -engine ctgostcp \  
-in data_file.enc -inform PEM -recip cert_file -inkey key_file \  
-out decrypt.data_file
```

При необходимости запрос пароля доступа к закрытому ключу будет произведен штатными средствами (например, через интерфейс СКЗИ КриптоПро CSP).

Работа с хэшами

В данном разделе рассматриваются хэш-операции с поддержкой ГОСТ алгоритмов. Операции хэширования производятся с помощью команды **dgst**.

Некоторые примеры, приводимые ниже, представлены в файле `crypto-tests.{bat|sh}`, поставляемые в дистрибутиве продукта «Trusted TLS 3».

Вычисление хэша

Непосредственное вычисление хэша по ГОСТ Р 34.11-94 можно произвести по команде

```
>openssl dgst -md_gost94 -engine ctgostcp -out data_file.hash data_file
```

Для вычисления хэша по ГОСТ Р 34.11-2012 (256 или 512 бит) вместо `-md_gost94` необходимо указать `-md_gost12_256` или `-md_gost12_512` соответственно.

Если определена переменная окружения **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine-модулей, то указание опции **-engine** не требуется.

Так как выходное значение хэша в hex-формате, а согласно спецификации хэш должен представлять little-endian число, то байты вывода хэша должны быть реверсированы.

Вычисление HMAC на основе ГОСТ Р 34.11-94 производится командой

```
>openssl dgst -md_gost94 -mac hmac -macopt key: 32bytes_of_key data_file
```

Если ключ содержит NUL байты, то вместо **key** нужно использовать **hexkey**.

Ниже приведен пример на вычисление GOST 28147 MAC

```
>openssl dgst -mac gost-mac -macopt key:32bytes_of_key data_file
```

Подпись хэша

Подписание хэша можно произвести по команде

```
>openssl dgst -engine ctgostcp -sign key_file -out data_file.sgn data_file
```

В приведенном примере вычисляется хэш данных из файла **data_file**, подписывается полученное значение хэша на основе файла закрытого ключа **key_file** и сохраняется в файле **data_file.sgn**.

Если определена переменная окружения **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine`-ов, то указание опции **-engine** не требуется.

Проверка подписи хэша

Проверку подписанного значения хэша можно произвести по команде

```
>openssl dgst -engine ctgostcp -verify pubkey-file -signature data_file.sgn data_file
```

В приведенном примере для проверки подписи хэша, хранящейся в файле **data_file.sgn**, используются хэшируемые данные из файла **data_file** и значение публичного ключа подписчика **pubkey-file**.

Если определена переменная окружения **OPENSSL_CONF**, указывающая на конфигурационный файл, в котором определено расположение требуемых для использования engine-модулей, то указание опции **-engine** не требуется.

Шифрование данных

В данном разделе рассматриваются примеры симметричного шифрования на основе ГОСТ 28147-89.

Шифрование данных методом ГОСТ 28147-89 в режиме CFB на основе пароля и кодирование результата в Base64 производится командой

```
>openssl enc -gost89 -e -a -out data_file.cfb.enc -in data_file -k passphrase
```

Шифрование данных методом ГОСТ 28147-89 в режиме CNT на основе пароля и кодирование результата в Base64 производится командой

```
>openssl enc -gost89-cnt -e -a -out data_file.cnt.enc -in data_file -pass "pass:passphrase"
```

Base64-декодирование и расшифрование данных методом ГОСТ 28147-89 в режиме CFB на основе пароля производится командой

```
>openssl enc -gost89 -d -a -out cfb.data_file -in data_file.cfb.enc -k passphrase
```

Использование опций **-K** и **-iv** в команде **enc** позволяет указать явно ключ шифрования и вектор инициализации в шестнадцатеричном виде. Вектор инициализации может быть рассчитан при отсутствии опции **-iv**, если задан пароль через опции **-k** или **-pass**.

Измерение производительности

Тестирование производительности симметричного шифрования можно произвести командой

```
>openssl speed -evp gost89
```

The 'numbers' are in 1000s of bytes per second processed.

Type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
GOST 28147-89	9819.40k	20333.46k	27940.37k	30034.34k	30351.13k

или

```
>openssl speed -evp gost89-cnt
```

SSL/TLS клиент

В данном разделе приводятся примеры доступа к защищенным ресурсам с использованием шифросюит (CipherSuite): TLS_GOSTR341001_WITH_28147_CNT_IMIT (0x0081), TLS_GOSTR341001_WITH_NULL_GOSTR3411 (0x0083) и TLS_GOSTR341112_256_WITH_28147_CNT_IMIT (0xFF85). В libssl данным шифросюитам соответствуют следующие идентификаторы:

- GOST2001-GOST89-GOST89 используется для аутентификации и обмена ключами на основе ГОСТ Р 34.10-2001, вычисления MAC на основе ГОСТ 28147-89 и шифрования данных по ГОСТ 28147-89.
- GOST2001-NULL-GOST94 используется для аутентификации и обмена ключами на основе ГОСТ Р 34.10-2001, вычисления HMAC на основе ГОСТ Р 34.11-94 и без шифрования данных.
- GOST2012-GOST89-GOST89 используется для аутентификации и обмена ключами на основе ГОСТ Р 34.10-2012, вычисления MAC на основе ГОСТ 28147-89 и шифрования данных по ГОСТ 28147-89.

Чтобы протестировать https-сервер `servername` с включенной клиентской авторизацией, необходимо выполнить команду

```
>openssl s_client -engine ctgostcp -connect servername:443 -cert cert_file -key key_file
```

Команда принимает клиентский сертификат **cert_file** и закрытый ключ из файла **key_file**. При отладке клиентской авторизации может оказаться полезной опция `-prexit`, выдающей список сертификатов УЦ, обслуживаемых сервером и др.

Перед вызовом в переменной окружения `OPENSSL_CONF` необходимо указать путь к конфигурационному файлу с engine-модулем `ctgostcp`.

После запуска клиента можно запросить с сервера требуемый файл, введя с консоли, например,

```
GET /<перевод строки><перевод строки>
```

SSL/TLS сервер

Для старта https сервера можно выполнить команду

```
>openssl s_server -engine ctgostcp -accept 443 -www -WWW -cert cert_file -key key_file
```

В данном примере на порту 443 стартует эмуляция простого web-сервера с расположением запрашиваемых страниц относительно текущего каталога. При старте

задается файл сертификата **cert_file** сервера и его закрытого ключа **key_file**. Запрашиваемые с сервера файлы должны быть расположены относительно каталога запуска утилиты.

Перед вызовом в переменной окружения OPENSSL_CONF необходимо указать путь к конфигурационному файлу с engine-модулем ctgostcp.

Перечень команд утилиты **ctgostcp_util**

В составе продукта присутствует утилита **ctgostcp_util**. Она позволяет:

- сгенерировать ключевой файл для существующего в системе контейнера закрытого ключа, расположенного в хранилище **КриптоПро CSP**;
- проверить на корректность ключевой файл на предмет существования в системе соответствующего контейнера закрытого ключа, расположенного в хранилище **КриптоПро CSP**, и доступа к нему по указанному паролю.

Формат использования **ctgostcp_util**:

ctgostcp_util <mode> [-option1 value1] [-option2] [...]

Для первого случая <mode> должно принимать значение **generate**, для второго – **verify**.

Для режима **generate** допустимо использование следующих параметров:

- -container <name> - имя контейнера (требуется при отсутствии опции -certfile)
- -certfile <certfile> - имя файла сертификата (требуется при отсутствии опции -container)
- -store user|machine|all - тип хранилища (по умолчанию: all)
- -keyspec sign|exch|auto - тип ключа (по умолчанию: auto)
- -passin <arg> - источник пароля
- -savepass - флаг сохранения пароля в ключевом файле
- -out <keyfile> - имя генерируемого ключевого файла (по умолчанию: stdout)
- -keyform PEM|DER - формат генерируемого ключевого файла (по умолчанию: PEM)

Опция **-container** требуется для указания имени контейнера закрытого ключа, для которого при наличии опции **-out** будет создан файл <keyfile> в формате PEM или, если это указано в опции **-keyform**, в формате DER.

При отсутствии опции **-container** требуется указать имя <certfile> файла сертификата, по которому будет произведен поиск соответствующего ему контейнера закрытого ключа.

Опция **-store** позволяет сузить диапазон поиска контейнера при указании типа хранилища.

При указании значения, отличного от **auto**, для опции **-keyspec** производится проверка на соответствие типу ключа.

Флаг **-savepass** указывает на необходимость сохранения в ключевом файле пароля доступа к контейнеру закрытого ключа. В этом случае должна быть определена опция **-passin**.

Согласно официальной документации проекта OpenSSL (см. http://www.openssl.org/docs/apps/openssl.html#PASS_PHRASE_ARGUMENTS) опция **passin** может принимать следующие значения:

- **pass:password**
В данной форме конкретный пароль **password** указывается непосредственно в командной строке утилиты.
- **env:var**
В данной форме указывается, что пароль требуется взять из переменной окружения **var**.
- **file:pathname**
В качестве **pathname** может быть указан файл, именованный канал или устройство. Первая строка его содержимого должна содержать значение пароля.
- **stdin**
В данном случае пароль будет затребован со стандартного устройства ввода
- **fd:number**
Данная форма позволяет получить пароль из источника, заданного номером **number** дескриптора файла.

Для режима **verify** допустимо использование следующих параметров:

- **-passin <arg>** - источник пароля (возможные значения см. в режиме [generate](#))
- **-in <keyfile>** - имя существующего ключевого файла
- **-keyform PEM|DER** - формат генерируемого ключевого файла (по умолчанию: PEM)

Техническая поддержка

ООО «Цифровые технологии»

Почтовый адрес:	424033, Россия, Республика Марий Эл, г. Йошкар-Ола, а/я 67
Тел.	+7 (8362) 33-70-50
Факс	+7 (8362) 33-70-50
Веб-сайт	http://www.trusted.ru/support/
Электронная почта:	
- общие вопросы	info@trusted.ru
- технические вопросы	support@trusted.ru