Ред База Данных

Версия 3.0

Функции шифрования

Содержание

Глава 1

Функции шифрования

1.1 Основные термины

Криптопровайдер — внешнее программное обеспечение, осуществляющее функции хэширования, шифрования, криптографической защиты.

Криптоплагин — библиотека, которая обеспечивает взаимодействие между криптопровайдером и сервером или утилитами сервера. Каждый криптоплагин предназначен для взаимодействия с определенным криптопровайдером.

1.2 Общие настройки

CryptoPlugin

Параметр `CryptoPlugin` определяет имя криптоплагина, который будет использоваться сервером для "общения" с Крипто-Про. Параметр имеет строковый тип. По умолчанию используется криптоплагин `crypto_api`, который размещен в каталоге `plugins`.

```
CryptoPlugin = Crypto_API
```

Криптоплагин реализует доступ к функциям криптопровайдера Крипто-Про. Он должен возвращать серверу указатели на экземпляры классов, связующих криптоплагин с криптопровайдером.

Задачами криптоплагина являются:

1. вычисление хэша;
2. заполнение буфера случайными числами;
3. шифрование данных (асимметричное и симметричное);
4. дешифрование данных (асимметричное и симметричное);
5. экспорт и импорт ключей;
6. генерация ключей для симметричного шифрования;
7. генерация пары ключей для асимметричного шифрования;
8. проверка ЭЦП;
9. проверка сертификата, построение цепочки сертификации до доверенного УЦ, проверка сертификата по COC (список отзыва сертификатов);
10. подпись данных.

Для конфигурирования криптоплагина в файле конфигурации `plugins.conf` добавлена специальная секция:

```
Plugin = CryptoAPI {
    Module = $(root)/plugins/CryptoAPI
    Config = CryptoAPI_config
}
Config = CryptoAPI_config {
    TracePlugin = 0
}
```

ProviderName

Задаёт имя или идентификатор используемого сервером криптопровайдера. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется провайдер «GOST R 34.10-2012 Signature with Diffie-Hellman Key Exchange».

```
ProviderName = 81
```

HashMethod

Задаёт название или идентификатор алгоритма хэширования, используемого сервером. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется алгоритм хэширования, заданный в ГОСТ Р 34.11-2012.

```
HashMethod = 32802
```

SymmetricMethod

Задаёт название или идентификатор алгоритма симметричного шифрования, используемого сервером. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется алгоритм шифрования, заданный в ГОСТ 28147-89.

```
SymmetricMethod = 26142
```

Алгоритмы хэширования зависят от используемого криптоплагина, подробнее см. п. 1.7.

Доступные алгоритмы и провайдеры

Таблица 1.1 — Доступные алгоритмы и провайдеры

ID	Название
Алгоритмы хэширования	
32798	ГОСТ Р 34.11/34.10-2001
32801	ГОСТ Р 34.11-2012/34.10-2012 256 бит
32802	ГОСТ Р 34.11-2012/34.10-2012 512 бит
Алгоритм симметричного шифрования	
26142	ГОСТ 28147-89
Провайдеры	
75	GOST R 34.10-2001 Signature with Diffie-Hellman Key Exchange
80	GOST R 34.10-2012 (256) Signature with Diffie-Hellman Key Exchange
81	GOST R 34.10-2012 (512) Signature with Diffie-Hellman Key Exchange

1.3 Шифрование сессии

Алгоритмы, выполняющие шифрование данных для разных целей, хорошо известны на протяжении многих лет. Единственной типичной проблемой остается то, где можно получить секретный

ключ, который будет использоваться этим алгоритмом. К счастью, для шифрования сетевого трафика есть одно хорошее решение - уникальный ключ шифрования может быть сгенерирован плагином аутентификации. По крайней мере, плагин SRP может создать такой ключ. К тому же этот ключ устойчив к атакам, включая атаку посредника (man-in-the-middle). Однако, если плагин аутентификации не предоставляет такой ключ, можно добавить псевдоплагин в список параметров `AuthClient` и `AuthServer` для создания ключей, что-то вроде пары ассиметричных закрытых/открытых ключей.

Шифрование сессии управляется двумя параметрами в файле `firebird.conf`: `WireCrypt` и `WireCryptPlugin`. Первый параметр включает/отключает шифрование. Второй же задает плагин шифрования. По умолчанию это `Arc4 (Alleged RC4)` - реализация шифра RC4. Другим возможным значением параметра `WireCryptPlugin` является `Wire_WinCrypt`. Если аутентификация в режиме SRP, то используется криптографический ключ, который делает шифрование сессии безопасным без необходимости обмена ключами между сервером и клиентом явно.

Плагин `Arc4` является «встроенным» в клиентскую библиотеку `fbclient.dll`. Однако можно получить или написать другой плагин защиты канала передачи данных.

Плагин `Wire_WinCrypt` использует параметры `ProviderName` и `SymmetricFactory` из `firebird.conf`. Стоит учитывать, что не все провайдеры и фабрики совместимы. Для шифрования используется `Crypto API`.

Плагины должны реализовывать специальные интерфейсы. Все эти интерфейсы применяют схему подсчета указателей (reference counting). Интерфейсы объявлены в файле `IdlFbInterfaces.h` каталога `include`. Полная инструкция по написанию плагинов выходит за рамки этого документа.

Все плагины находятся в папке `plugins` корневой папки установки сервера. Файл `plugins.conf` служит для настройки плагинов. Если в нем не указаны настройки для плагина, то применяется конфигурация по умолчанию. Файл конфигурации состоит из двух типов блоков:

```
Plugin = <имя плагина> {
   Module = <путь до библиотеки с плагином>
   Config = <имя конфигурации>
}
Config = <имя конфигурации> {
   <ключ> = <значение>
   ...
}
```

1.4 Шифрование базы данных

В Ред Базе Данных существует возможность поддержки шифрования базы данных. Не весь файл базы данных шифруется: только страницы данных, индексов и `blob`.

Для того чтобы сделать шифрование базы данных возможным необходим плагин шифрования базы данных. Ред База Данных предоставляет плагины `Crypto_Api` и `RdbCrypt`. Также плагин можно написать самостоятельно.

> Пример плагина шифрования в `examples/dbcrypt` не производит реального шифрования, а просто демонстрирует, как можно написать этот плагин.

Необходимо принимать во внимание следующее. Существует два основных применения шифрования БД:

- чтобы предотвратить утечку важных (секретных) данных, если сервер БД взломан/физически украден.
- когда база данных поставляется вместе с каким-либо приложением, использующим подобного рода данные (т.е. хоть БД и есть на руках у злоумышленника, но она зашифрована, а секретные данные можно получить только через приложение).

В первом случае мы можем доверять серверу базы данных, что он не модифицирован для кражи ключей, передающихся плагину безопасности - то есть мы ожидаем, что этот ключ не будет отправлен на неподходящий сервер. Во втором случае сервер может быть каким-то образом модифицирован для кражи ключей (если они передаются из приложения в плагин через код сервера) или даже данных (например, как крайний случай, модифицированный сервер выгружает блоки данных из кэша, где они не зашифрованы). Поэтому плагин должен убедиться, что файлы сервера, содержащие исполнимый код, не модифицированы, и приложение перед отправкой ключа плагину должно быть уверено в подлинности плагина (например, потребовать от него цифровую подпись). Если используется сетевой доступ к серверу, хорошим тоном будет убедиться, что сеть зашифрована или использовать собственное шифрование ключа. Все эти действия должны выполняться в плагине и приложении, работающем с ним. Таким образом алгоритм шифрования блока базы данных сам по себе может оказаться самой простой частью плагина, особенно когда для него используется какая-нибудь стандартная библиотека.

Основная проблема с шифрованием базы данных состоит в том, как хранить секретный ключ. Ред База Данных предоставляет помощника для передачи этого ключа от клиента, но это вовсе не означает, что хранение ключей на клиенте является лучшим способом: это не более чем одна из возможных альтернатив. Хранение ключей на том же диске, что и база данных является очень плохим вариантом.

Для эффективного разделения шифрования и доступа к ключу, плагин шифрования базы данных разделён на две части: само шифрование и держатель секретного ключа. Плагин шифрования имеет дело с фактическим шифрованием, тогда как плагин для хранения ключа решает вопросы, связанные с предоставлением ключа безопасным способом. Этот плагин может быть получен из приложения или загружен каким-либо другим способом (вплоть до использования флеш-устройства, вставленного в сервер при запуске сервера Ред Базы Данных). Для задания плагина держателя секретного ключа в файле конфигурации нужно определить значение параметра `KeyHolderPlugin`.

Процесс шифрования включается следующим способом:

```
ALTER DATABASE ENCRYPT WITH <имя плагина> [KEY <имя ключа шифрования>]
```

Шифрование начинается сразу после этого оператора и будет выполняться в фоновом режиме. Нормальная работа с базами данных не нарушается во время шифрования.

Необязательное предложение `KEY` позволяет передать имя ключа для плагина шифрования. Что делать с этим именем ключа решает плагин.

Для дешифрования базы данных выполните:

```
ALTER DATABASE DECRYPT
```

1.4.1 Шифрование базы данных с использованием плагина Crypto_Api

Для его использования плагина `Crypto_Api` нужно указать `"Crypto_Api"` в операторе `ALTER DATABASE` и параметре конфигурации `KeyHolderPlugin`.

Для шифрования базы данных средствами плагина `Crypto_Api` должен быть доступен хотя бы один ключевой контейнер КриптоПро. Имя используемого ключевого контейнера указывается в параметре `KEY` оператора `ALTER DATABASE`. Если ключевой контейнер защищен пином, то предварительно нужно добавить его в файл конфигурации плагина в формате: `<Имя_контейнера> = <пин>`.

Для каждой базы данных, зашифрованной плагином `Crypto_Api` создается уникальный публичный ключ. Публичные ключи хранятся в директории `KeyDirectory` указанной в конфигурации плагина `Crypto_Api` в файле `plugins.conf`. В качестве имени ключевого файла используется имя ключевого контейнера КриптоПро, который был указан в параметре `KEY`. Публичные ключи можно безопасно передавать по любому каналу связи, так как они зашифрованы и могут быть использованы только совместно с оригинальным ключевым контейнером КриптоПро, использовавшимся при шифровании.

Для последующего использования зашифрованной базы данных в системе должен обязательно присутствовать оригинальный ключевой контейнер КриптоПро и сгенерированный уникальный публичной ключ для конкретной базы данных. Дешифрование базы данных невозможно при наличии только ключевого контейнера или только публичного ключа, наличие обоих обязательно.

Для дешифрования базы данных выполните:

```
ALTER DATABASE DECRYPT
```

Если в системе присутствует оригинальный ключевой контейнер и публичный ключ для используемой базы данных, то база данных будет расшифрована и доступна без ключей шифрования.

Если после полного дешифрования базы данных не удалить предыдущий ключ шифрования (файл публичного ключа), то он будет использован повторно при следущем запросе на шифрование базы данных. Чтобы сгенерировать новый ключ шифрования, обязательно нужно удалить использовавшийся ранее из директории файлов ключей.

1.4.2 Шифрование базы данных с использованием плагина RdbCrypt

Плагин шифрования `RdbCrypt` основан на открытой реализации блочного алгоритма шифрования `AES-256`. `RdbCrypt` в использовании проще чем `Crypto_Api`, но менее надежен, потому что использует только один файл ключа.

Для использования RdbCrypt его нужно указать `"RdbCrypt"` в операторе `ALTER DATABASE` и в параметре конфигурации `KeyHolderPlugin`. После этого запроса будет сгенерирован файл ключа в директории, указанной в параметре `KeyHolderPlugin` конфигурации плагина с именем ключа, который использован в запросе. Если файл ключа уже существует, то он будет использован, а не сгенерирован новый.

Файл ключа не обазательно должен быть сгенерирован плагином, его можно создать или использовать любой файл, который не превышает размер 1000 байт. Например, можно использовать `key.txt` с текстом "секретный ключ".

Процесс дешифрования не отличается от `Crypto_Api`, кроме того что используется только один файл ключа, который должен присутствовать в директрии `KeyDirecotry` указанной в конфигурации плагина.

1.5 Контроль целостности

Контроль целостности программных и информационных ресурсов в Ред База Данных реализована на основе расчёта и последующей проверки хэш значений контролируемых объектов.

При сборке дистрибутива, после того, как сформированы все компоненты системы, запускается утилита формирования хэшей. Она хэширует файлы, указанные в параметрах, и помещает результат в файл хэшей.

При инсталляции и в процессе эксплуатации администратор может модифицировать список файлов, подлежащих контролю, и сгенерировать для них файл хэшей-эталонов (например, добавить к контролируемым файлам файл конфигурации). Для этого используется утилита `hashgen`, входящая в состав дистрибутива Ред База Данных подробнее см. п. 1.7.

При старте сервера, после того, как инициализируется криптопровайдер, производится считывание содержимого файла с хэшами. Далее, для каждого подконтрольного файла генерируется хэш с определенным для него алгоритмом и сверяется с эталонным значением хэша. При несоответствии сгенерированного и хранимого хэшей в журнал заносится запись с именем файла, имеющего неверную контрольную сумму и сервер прекращает работу.

Включение режима контроля целостности файлов сервера выполняется путем задания имени файла с хэшами в конфигурационном файле Ред База Данных (параметр `HashesFile`).

Кроме того, аналогично описанному выше, может также контролироваться целостность метаданных в конкретной базе данных. Для этого используется утилита `mint`, также входящая в состав дистрибутива Ред База Данных (см. п. 1.6).

1.6 Контроль целостности метаданных [1]

Контроль за целостностью метаданных в БД осуществляется с помощью утилиты `mint`. Эта утилита предназначена для извлечения и хэширования метаданных из баз данных, а также для проверки ранее полученного хэша метаданных. Таким образом, администратор может защитить структуру базы данных от изменений.

Утилита `mint` позволяет выбрать все метаданные из базы данных или только их часть, по заданной маске, хэшировать их и сохранить в файл. Также утилита может проводить проверку текущего состояния метаданных в базе путем повторной выборки данных и сравнения результата с ранее сохраненным.

Утилита имеет следующие команды и опции:

Таблица 1.2 — Команды и опции утилиты `mint`

Команды и опции	Описание
`-M (mode) {extract|check}`	Режим работы утилиты
`-o (object)`	Объекты для извлечения метаданных (системные и определенные пользователем, по умолчанию - все)
`-s (system)`	Извлекать метаданные системных объектов
`-m (mask)`	Маска для извлечения метаданных из БД
`-d (database)`	Имя базы данных, из которой будет производиться извлечение данных
`-u (user)`	Имя пользователя для присоединения к БД
`-p (password)`	Пароль пользователя для присоединения к БД
`-r (repository)`	Хранилище с ключами шифрования
`-P (pin)`	PIN (пароль) для закрытого ключа сертификата пользователя
`-R (repository-algorithm)`	Алгоритм шифрования для хранилища ключей
`-i (signature-file)`	Файл для сохранения|проверки хэша метаданных
`-I (sign-algorithm)`	Название алгоритма цифровой подписи
`-h (help)`	Отображает все опции `mint`
`-v (verbose)`	Подробный вывод

Существуют два режима работы утилиты – генерация контрольной суммы (`extract`) и проверка (`check`).

Если маска не задана, то выбираются все метаданные из базы. В маске можно использовать символ `%` - заменяет собой любое количество любых символов.

Например, генерация подписи для всех системных объектов (начинающихся с префикса RDB$) в базе данных `security3.fdb`:

[1] Для работы данной функции необходимо наличие криптопровайдера и криптоплагина (последний входит в поставку СУБД Ред База Данных). Криптопровайдер реализует функции шифрования и хэширования, а криптоплагин предоставляет унифицированный интерфейс между криптопровайдером и сервером СУБД Ред База Данных. В качестве криптопровайдера в настоящее время используется КриптоПро CSP 4.0

```
mint -M extract -m RDB$% -d ../security3.fdb -u sysdba -p masterkey -r test -R
"Crypto Pro GOST R 34.10-2012 KC1 CSP" -i sign -I AT_SIGNATURE
```

проверка подписи для этих объектов:

```
mint -M check -m RDB$% -d ../security3.fdb -u sysdba -p masterkey -r test -R
"Crypto Pro GOST R 34.10-2012 KC1 CSP" -i sign -I AT_SIGNATURE
Signature verification complete successfully
```

1.7 Контроль целостности файлов сервера [2]

Контроль за файлами сервера означает, что для всех критически важных файлов сервера (бинарные файлы, файлы конфигурации, база данных безопасности `security3.fdb` и т. д.) может быть вычислен и проверен хэш. Для этого используется утилита `hashgen`, входящая в состав дистрибутива. Она использует алгоритмы, предоставляемые криптопровайдером КриптоПро.

Файл с контрольными суммами (с хэшами) всех защищаемых файлов, а также файл конфигурации Ред База Данных, в котором указан путь к файлу с хэшами, должны быть защищены с помощью организационно-технических мер (используется аппаратная защита, исключающая его модификацию и повреждение) или применяются регламентные меры для контроля их целостности.

Для того, чтобы включить контроль целостности файлов сервера, необходимо указать имя файла, содержащего контрольные суммы (хэши) защищаемых файлов сервера в конфигурационном файле Ред База Данных `firebird.conf`. Имя этого файла задается параметром `HashesFile`. Если задано значение этого параметра, то каждый раз при запуске сервера и регулярно в процессе работы СУБД в соответствии со значением параметра `IntegrityCheckInterval` файла конфигурации `firebird.conf`, происходит проверка целостности файлов сервера (только файлов на внешнем носителе). При этом файл с хэшами считывается один раз при первой проверке.

При старте сервера, после того, как инициализируется криптопровайдер, производится считывание содержимого файла с хэшами. Далее, для каждого подконтрольного файла генерируется хэш с определенным для него алгоритмом и сверяется с эталонным значением хэша.

Файл хэшей должен содержать строки вида:

```
<хэш>:<алгоритм хэширования>:<имя файла>:<опция>
```

Где `<опция>` принимает значения `S` и `M` для файла на внешнем носителе и в оперативной памяти соответственно.

Если какой-либо из хэшей не совпал, соответствующий файл не найден или произошла ошибка при проверке, сервер делает соответствующую запись в `firebird.log` и завершает работу. В зависимости от параметра конфигурации `IntegrityShutdownAttempts` сервер совершает несколько (или ноль) попыток прекратить свою работу безопасными средствами. Если сервер не удалось выключить после заданного количества попыток, то выполняется функция `exit(FINI_ERROR);`.

Генерация файла с хэшами производится во время сборки дистрибутива специальной утилитой `hashgen`. В последующем администратор может пересоздать хэш отдельно взятого файла.

Формат запуска утилиты имеет следующий вид:

```
hashgen generate {-S|-M <PID>} <алгоритм хэширования> <файл_сервера_1>
[<файл_сервера_2>] ...
```

Входные параметры утилиты – имя хэшируемого файла, алгоритм хэширования и ключи `-S` (или `--storage`) и `-M` (или `--memory`) для генерации хэшей файла на внешнем носителе и в оперативной памяти соответственно.

[2] См. примечание п. 1.6.

В частности, опция `-M` применяется для всех библиотек и исполняемых файлов, поддерживающих верификацию оперативной памяти процесса. В качестве информации, подлежащей хэшированию, выступает образ исполняемого файла или библиотеки в PE-формате для Windows и в ELF-формате для Linux. хэшируются секции образа, изменение которых не предусмотрено. Ниже представлен обобщенный список, некоторые из секций в котором специфичны только для конкретного формата:

1. секция кода;
2. секция константных данных;
3. секции экспорта/импорта;
4. секция ресурсов.

Также утилитой `hashgen` можно производить проверку ранее созданных хэшей:

```
hashgen check [-P <PID>] <файл_хэшей_1> [<файл_хэшей_2>] ...
```

В качестве основного параметра выступают имена файлов, содержащие хэши.

Если хэш был сгенерирован для файла в памяти (т.е. с опцией `-M <PID>`), то при проверке нужно указывать ключ `-P <PID>`. Иначе проверка не будет выполнена, но утилита сообщит об успешном завершении.

Пример:

Генерация контрольной суммы алгоритмом ГОСТ Р 34.11-2012 для файла `security3.fdb` и сохранение в файл `test.sign`:

```
hashgen generate -S 32802 ../security3.fdb>test.sign
```

Пример генерации хэша из оперативной памяти:

```
hashgen generate -M 1234 32802 rdbserver.exe > test.sign
```

Проверка хэшей:

```
hashgen check test.sign
../security3.fdb: Success
```

Контрольная сумма в файле `test.sign` совпадает с контрольной суммой исходного файла.

Глава 2

Настройка КриптоПро (на примере версии 4.0)

2.1 Настройка в операционной системе Windows

2.1.1 Общие настройки

1. Скачайте дистрибутив КриптоПро с официального сайта КриптоПРО в разделе загрузка (http://www.cryptopro.ru/downloads). Доступ к скачиванию обеспечивается после регистрации на сайте.
2. Установите КриптоПро.
3. Настройте провайдер через Панель управления->КриптоПро CSP. На вкладке *Алгоритмы* настраиваются параметры для криптопровайдера (рисунок 2.1).

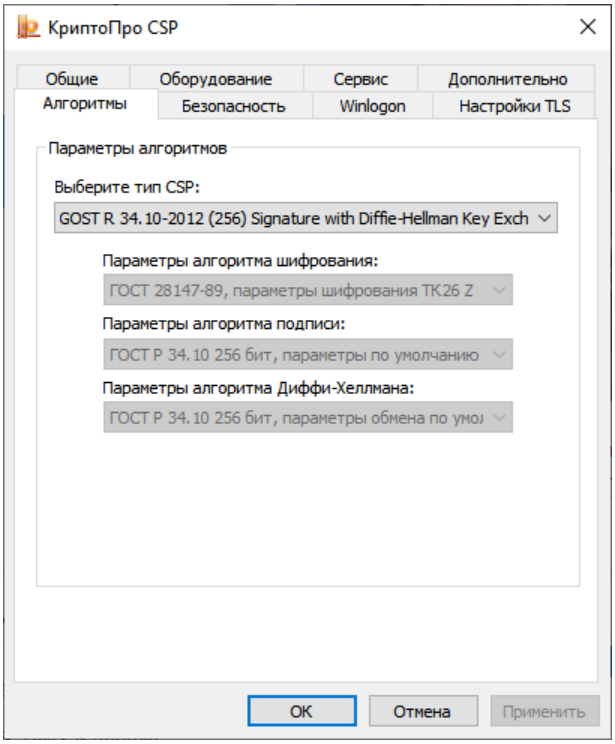


Рисунок 2.1 — Настройка алгоритмов

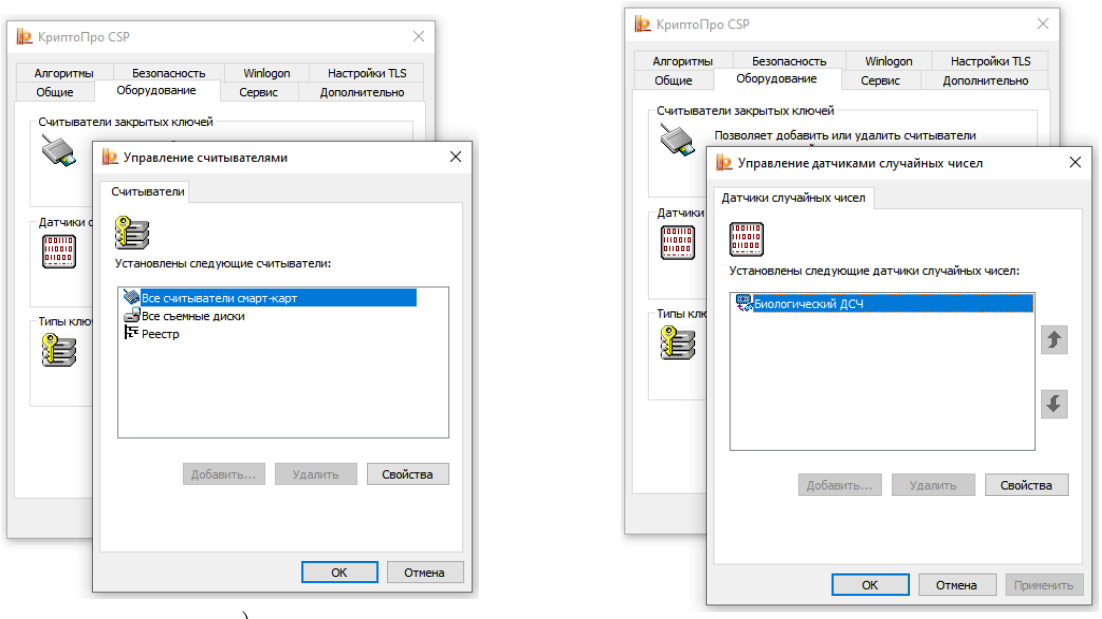
- Название криптопровайдера – `GOST R 34.10-2012 Signature with Diffie_Hellman Key Exchange`. Используется в параметре `ProviderName` файла конфигурации `firebird.conf`;
- Параметры алгоритма шифрования – название алгоритма шифрования по умолчанию. Используется в параметре `SymmetricMethod` файла конфигурации `firebird.conf`;
- Параметры алгоритма подписи – название алгоритма ЭЦП по умолчанию, аналогично использование псевдонима `AT_SIGNATURE` как значение параметра для утилиты `mint`;

- Параметры алгоритма Диффи-Хелмана – название алгоритма, который будет использоваться по умолчанию для обмена сессионными ключами, аналогично использованию псевдонима `AT_KEYEXCHANGE` как значение параметра для утилиты `mint`.

2.1.2 Создание контейнеров с закрытыми ключами

Сессионные ключи генерируются плагином автоматически на время сессии. Для более безопасного обмена ими и аутентификации с использованием сертификата желательно использовать секретные ключи, которые хранятся в контейнерах.

1. Контейнеры сохраняются на считывателях, соответственно, необходимо настроить считыватели на вкладке оборудование (рисунок 2.2 а)).
2. Необходимо установить биологический датчик случайных чисел на вкладке оборудование (рисунок 2.2 б)).



а) б)

Рисунок 2.2 — Настройка считывателей и датчиков случайных чисел

3. Для генерирования секретных ключей можно воспользоваться средством КриптоПро `<каталог_установки_КриптоПро>\csptest.exe`:

```
csptest -keyset -provtype 81 -newkeyset -container <имя контейнера с секретным ключом>
```

4. Необходимо создавать ключи без паролей, так как при аутентификации используется режим `CRYPT_SILENT` (чтобы не нарушать работу сервера).

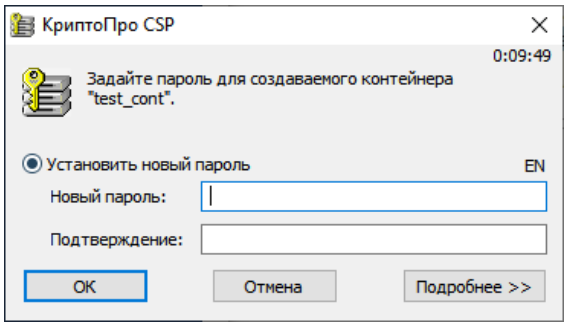


Рисунок 2.3 — Запрос установки пароля

5. При создании ключа следует учитывать разграничение доступа среди пользователей Windows, то есть ключ, созданный одним пользователем, не будет доступен другому. Если сервер работает в режиме службы (то есть от имени системного пользователя), то необходимо, чтобы владельцем контейнера с серверными ключами была сама операционная система. Для этого необходимо при создании серверного контейнера указать опцию `-machinekeys`:

```
csptest -keyset -provtype 81 -newkeyset -container <имя контейнера> -machinekeys
```

6. Ключи хранятся в следующем разделе реестра:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Crypto Pro\Settings\USERS\SID пользователя\Keys\<имя
контейнера>
```

Пример создания ключевой пары

```
csptest -keyset -newkeyset -container test_keyq
CSP (Type:80) v4.0.9014 KC1 Release Ver:4.0.9842 OS:Windows CPU:IA32
FastCode:READY:AVX.
AcquireContext: OK. HCRYPTPROV: 8411672
GetProvParam(PP_NAME): Crypto-Pro GOST R 34.10-2012 Cryptographic Service Provider
Container name: "test_keys"
Signature key is not available.
Attempting to create a signature key...
a signature key created.
Exchange key is not available.
Attempting to create an exchange key...
an exchange key created.
Keys in container:
signature key
exchange key
Extensions:
OID: 1.2.643.2.2.37.3.9
PrivKey: Not specified - 17.08.2018 19:19:40 (UTC)
OID: 1.2.643.2.2.37.3.10
PrivKey: Not specified - 17.08.2018 19:19:59 (UTC)
Total: SYS: 0,515 sec USR: 0,359 sec UTC: 121,656 sec
[ErrorCode: 0x00000000]
```

2.1.3 Получение сертификата

Для получения сертификата в ОС Windows с помощью тестового центра КриптоПро выполните следующие действия:

1. Сформируйте запрос на получение сертификата с помощью утилиты `cryptcp` и ее команды `creatrqst`.

```
-creatrqst -dn <RDN> [-provtype <N>] [-provname <CSP>] [-SMIME]
[-nokeygen|-exprt] [-keysize <n>] [-hashAlg <OID>] [-ex|-sg|-both] [-ku|-km]
[-cont <имя>] [-silent] [-pin <пароль>|-askpin] [-certusage <OIDs>] [-der] [-ext
<расширение>] <имя файла>
```

   При генерации запроса для клиентского сертификата необходимо указать заранее созданный клиентский контейнер с ключевой парой. Для этого используется опция `-nokeygen`. Например:

```
cryptcp -creatrqst -nokeygen -cont test_keys -provtype 81 -dn 'CN=tester'
test.req
```

   Подробнее о командах и входных параметрах этой утилиты см. руководство по приложению командной строки `CryptCP` по адресу: https://www.cryptopro.ru/sites/default/files/products/cryptcp.

2. Откройте в браузере ссылку http://www.cryptopro.ru/certsrv (тестовый удостоверяющий центр КриптоПро).
3. Нажмите «Отправить готовый запрос PKCS#10 или PKCS#7 в кодировке Base64».
4. Вставьте в поле «Base-64-шифрованный запрос сертификата» содержимое файла `test.req` и нажмите кнопку «Выдать».
5. Сохраните файл по ссылке «Загрузить сертификат». Выгрузку сертификата необходимо проводить в формате Base64.
6. Установите, полученный от УЦ сертификат в указанный ключевой контейнер:

```
certmgr -inst -store uMy -file certnew.cer -cont test_keys
```

Также в ОС Windows возможно получение сертификата с помощью тестового центра КриптоПро http://www.cryptopro.ru/certsrv/ без самостоятельного создания запроса.

Для этого нажмите «Сформировать ключи и отправить запрос на сертификат» и заполните некоторые данные.

Выданные сертификаты можно посмотреть в окне «Сертификаты» (Пуск⟶КРИПТО-ПРО⟶Сертификаты).

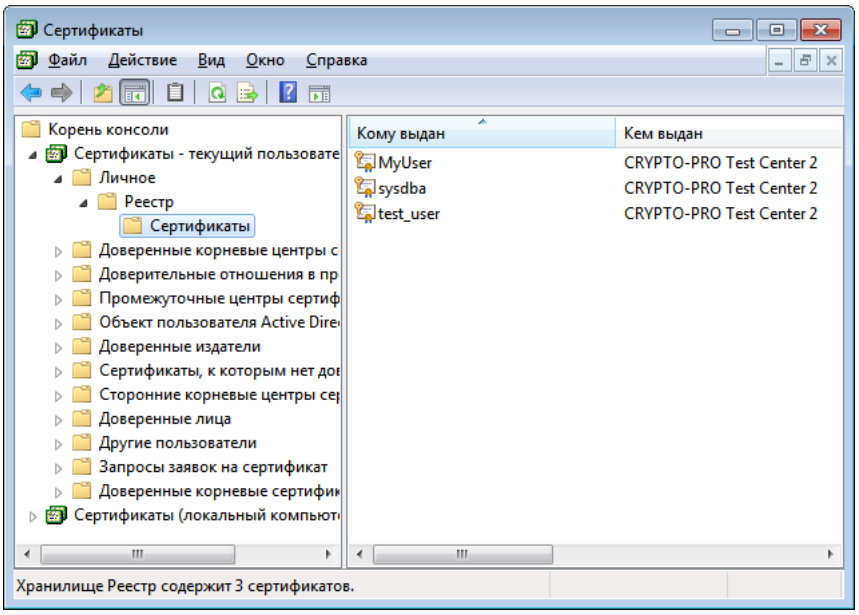


Рисунок 2.4 — Просмотр сертификатов

2.2 Настройка в операционной системе Linux

2.2.1 Общие настройки

1. Архив с программным обеспечением можно загрузить с официального сайта КриптоПРО в разделе загрузка (http://www.cryptopro.ru/downloads): `linux-ia32.tgz, linux-amd64.tgz`.

   > По умолчанию при скачивании с сайта КриптоПро выдается лицензия на три месяца

2. Распакуйте архив и перейдите в распакованную папку.
3. Установите основные пакеты:

```
rpm -i lsb-cprocsp-base-X.X.X-X.rpm
rpm -i lsb-cprocsp-rdr-X.X.X-X.rpm
rpm -i lsb-cprocsp-capilite-X.X.X-X.rpm
rpm -i lsb-cprocsp-kc1-X.X.X-X.rpm
```

На ОС, основанных на Debian (Debian/Ubuntu) сначала необходимо поставить (если не установлены) пакеты LSB из состава дистрибутива, а также пакет `alien`, который является штатным средством для установки `.rpm`:

```
apt-get install lsb-base alien lsb-core
```

Затем при помощи `alien` поставить необходимые пакеты CSP, например:

```
alien -kci ./lsb-cprocsp-base-X.X.X-X.rpm
alien -kci ./lsb-cprocsp-rdr-X.X.X-X.rpm
alien -kci ./lsb-cprocsp-capilite-X.X.X-X.rpm
alien -kci ./lsb-cprocsp-kc1-X.X.X-X.rpm
```

КриптоПро устанавливается по пути `/opt/cprocsp`.
По мере удовлетворения зависимостей, возможно, потребуется установить другие пакеты.

4. В Linux поддерживается провайдер `Crypto-Pro GOST R 34.10-2012 KC1 CSP`.
5. Настройте сервер «Ред База Данных» на работу с криптоплагином. Для этого в конфигурационном файле `firebird.conf` указываются следующие параметры:
   - `CryptoPlugin` – имя криптоплагина (значение по умолчанию `Crypto_API`)
   - `SymmetricMethod` – алгоритм, используемый для симметричного шифрования. Можно указывать название алгоритма или его идентификатор в криптопровайдере (`AlgId`). Идентификатор алгоритма можно узнать утилитами `mint` и `hashgen` при запуске без параметров (параметр `CryptoPlugin` в конфигурационном файле должен быть задан) или утилитой КриптоПро `csptest` с параметрами `-enum -info`. Если используется неанглоязычные названия алгоритмов, необходимо перекодировать их в 2-х байтный юникод, например `ГОСТ Р 34.11-2012` примет вид `%D0%93%D0%9E%D0%A1%D0%A2+%D0%A0+34.11-2012`
   - `HashMethod` – название или идентификатор алгоритма хэширования. Способ представления аналогичен `SymmetricMethod`.
   - `ProviderName` – название или идентификатор алгоритма криптопровайдера. Способ представления аналогичен `SymmetricMethod`.

2.2.2 Создание контейнера с закрытыми ключами

1. Ридеры (readers) — устройства размещения контейнеров (аппаратные токены, каталог для размещения файлов). Просмотр доступных ридеров:

```
$ csptest -enum -info -type PP_ENUMREADERS | iconv -f cp1251
```

   Ридер `HDIMAGE` размещается на `/var/opt/cprocsp/keys/<имя пользователя>/`.
2. Инициализация ридера `HDIMAGE` (под правами root):

```
cpconfig -hardware reader -add HDIMAGE store
```

3. Создадим контейнер с именем `test` в локальном ридере `HDIMAGE`:

```
$ csptest -keyset -provtype 81 -newkeyset -cont '\\.\HDIMAGE\test'
```

   При установленном пакете `cprocsp-rdr-gui-gtk` будет показано графическое окно, где предложат двигать курсором мыши. Если такой пакет не установлен, будет предложено ввести любые символы с клавиатуры. После показа окна будет предложено указать пароль на контейнер (можно указать пустой, тогда пароль запрашиваться не будет) и снова предложат двигать курсором мыши.

```
CSP (Type:81) v4.0.9006 KC1 Release Ver:4.0.9708 OS:Linux CPU:AMD64
FastCode:READY:AVX.
AcquireContext: OK. HCRYPTPROV: 6679219
GetProvParam(PP_NAME): Crypto-Pro GOST R 34.10-2012 KC1 CSP
Container name: "card"
Signature key is not available.
Attempting to create a signature key...
a signature key created.
Exchange key is not available.
Attempting to create an exchange key...
an exchange key created.
```

```
Keys in container:
signature key
exchange key
Extensions:
OID: 1.2.643.2.2.37.3.9
OID: 1.2.643.2.2.37.3.10
Total: SYS: 0,030 sec USR: 0,160 sec UTC: 22,910 sec
[ErrorCode: 0x00000000]
```

4. Ключи в Linux хранятся в `/var/opt/cprocsp/keys/<имя пользователя>/<имя контейнера>`.
5. Просмотр доступных контейнеров:

```
$ csptest -keyset -enum_cont -fqcn -verifyc
```

6. Удаление контейнера:

```
$ csptest -keyset -deletekeyset -cont '\\.\HDIMAGE\test'
```

2.2.3 Создание сертификата

Для получения сертификата в ОС Linux с помощью тестового центра КриптоПро выполните следующие действия:

1. Сформируйте запрос на получение сертификата с помощью утилиты `cryptcp` и ее команды `creatrqst`.

```
-creatrqst -dn <RDN> [-provtype <N>] [-provname <CSP>] [-SMIME]
[-nokeygen|-exprt] [-keysize <n>] [-hashAlg <OID>] [-ex|-sg|-both] [-ku|-km]
[-cont <имя>] [-silent] [-pin <пароль>|-askpin] [-certusage <OIDs>] [-der] [-ext
<расширение>] <имя файла>
```

При генерации запроса для клиентского сертификата необходимо указать заранее созданный клиентский контейнер с ключевой парой. Для этого используется опция `-nokeygen`. Контейнер может быть указан без спецификатора `HDIMAGE`. Например:

```
opt/cprocsp/bin/ia32/cryptcp -creatrqst -nokeygen -cont '\\.\HDIMAGE\test'
-provtype 81 -dn 'CN=tester' test.req
```

> Контейнер с ключевой парой, который используется для генерации запроса на получение клиентского сертификата, должен принадлежать тому же пользователю, для которого запрашивается сертификат.

Подробнее о командах и входных параметрах этой утилиты см. руководство по приложению командной строки `CryptCP` по адресу: https://www.cryptopro.ru/sites/default/files/products/cryptcp.

2. Откройте в браузере ссылку http://www.cryptopro.ru/certsrv (тестовый удостоверяющий центр КриптоПро).
3. Нажмите «Отправить готовый запрос PKCS#10 или PKCS#7 в кодировке Base64»
4. Вставьте в поле «Base-64-шифрованный запрос сертификата» содержимое файла `test.req` и нажмите кнопку «Выдать».
5. Сохраните файл по ссылке «Загрузить сертификат». Выгрузку сертификата необходимо проводить в формате Base64.

> Не все браузеры в ОС Linux могут поддерживать выгрузку сертификатов. Это зависит от настроек сервера центра сертификации.

6. Устанавливаем, полученный от УЦ сертификат, в указанный ключевой контейнер:

```
$ certmgr -inst -store uMy -file certnew.cer -cont '\\.\HDIMAGE\test'
```

7. Чтобы посмотреть сертификаты, введите команду:

```
$ certmgr -list
```

8. Чтобы удалить сертификат, введите команду:

```
$ certmgr -delete 1
$ certmgr -delete -all
```