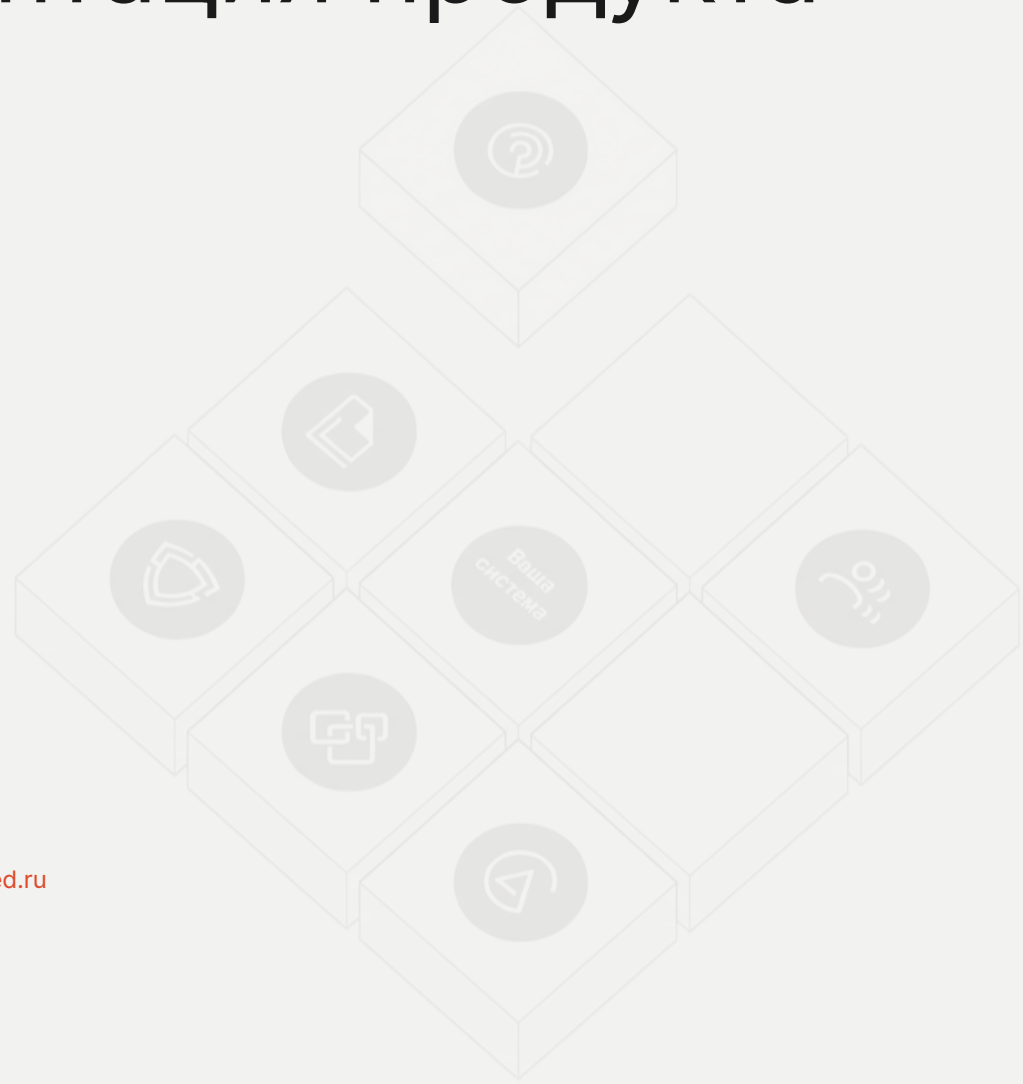
Цифровые технологии

Trusted Java

Документация продукта

Релиз v2.0



Исходная документация на Trusted.ru

Редакция от 13.11.2026

Правообладатель и разработчик
ООО «Цифровые технологии»

Содержание

Руководство администратора

- Общие сведения о продукте ... 3
- Установка продукта ... 8
- Удаление продукта ... 14
- Настройка связки Trusted Java и Tomcat под Linux ... 16
- Java Bridge: Интеграция Trusted Java и PHP ... 22
- Сервера приложений: интеграция Trusted Java и TrustedTLS ... 25
- Часто задаваемые вопросы ... 30

Руководство разработчика

- Программные интерфейсы ... 33
- Классы, реализующие ГОСТ алгоритмы ... 41

Открыть статью на сайте

Общие сведения о продукте

Программный продукт Trusted Java является средством криптографической защиты информации и представляет собой набор криптоалгоритмов, реализованных в соответствии с требованиями архитектур JSSE и JCE.

Провайдер JSSE обеспечивает организацию защищенного взаимодействия по протоколам SSL и TLS с использованием российских криптографических алгоритмов. Таким образом, он даёт возможность устанавливать безопасные интернет-соединения и обеспечивать защиту канала передачи данных. Кроме того, компонент включает функциональность для шифрования данных, аутентификации клиента и сервера и целостности сообщений.

Провайдер JCE является криптографическим расширением Trusted Java, т.е. позволяет реализовать российские алгоритмы криптографических преобразований для обеспечения работы с сертификатами, создания и проверки корректности электронной цифровой подписи и шифрования данных. ПО Trusted Java обеспечивает возможность поддержки российской криптографии в соответствии с положениями российского законодательства.

Для авторизации и обеспечения юридической значимости электронных документов при обмене ими между пользователями используются функции формирования и проверки электронно-цифровой подписи, для обеспечения конфиденциальности информации и контроля ее целостности – шифрование и имитозащита.

Также Trusted Java позволяет работать с сертификатами и запросами на сертификат, создавать и проверять корректность электронной цифровой подписи данных, шифровать данные, загружать Java апплеты по протоколу TLS на криптографических алгоритмах ГОСТ.

Используемые сертифицированные российские криптоалгоритмы предназначены для:

- авторизации и обеспечения юридической значимости электронных документов при обмене ими между пользователями (создание и проверка ЭЦП).
- обеспечения конфиденциальности и контроля целостности информации (шифрование и имитозащита).

Библиотека Trusted Java предназначена, прежде всего, для системных интеграторов и разработчиков прикладных и бизнес-систем для решения вопросов информационной безопасности и юридической значимости электронного документооборота. При работе с банковскими клиентсерверными системами, защищенными электронными торговыми площадками и другими бизнес-приложениями, написанными на Java, требуется поддержка шифрования, электронной цифровой подписи, строгой аутентификации в соответствии с положениями российского законодательства. ПО Trusted Java позволяет использовать эти возможности.

История версий

В сборке r507 доработано чтение сертификатов из хранилища. Ранее не возвращались сертификаты с алгоритмом публичного ключа ГОСТ 2012 (256 - 512). Использование:

```
CryptoProCSPCertStore storeCsp = new CryptoProCSPCertStore();
Collection certificates = storeCsp.getInstance("CurrentUser/ROOT",
null).getAllCertificates();
```

В сборке r506 добавлена возможность проверки подписи с отключением/включением сортировки подписанных атрибутов (по умолчанию они не сортируются. Использование:

```
SignerInformation si;
...
si.setNeedSortSignedAttributes(true);
si.verify(cert, PROV_NAME))
```

В сборке r505 добавлена поддержка алгоритмов ГОСТ 2012 для XML. В версии Trusted Java 2.0 (сборка r504 и выше) добавлена поддержка работы с ключами и алгоритмами ГОСТ Р 34.11-2012/ГОСТ Р 34.10-2012 (только с КриптоПро CSP 4.0).

В версии Trusted Java 2.0 изменилась система лицензирования. Теперь на серверных версиях ОС Windows для операций подписи и шифрования, а также для использования клиентских сокетов достаточно клиентской лицензии. Проверка подписи стала бесплатной, на всех платформах. Теперь полученный лицензионный ключ действителен только для одной платформы ОС и не подходит к другой. Поэтому следует пользователям, переходящим на новую версию, обновить или приобрести под другую платформу лицензионные ключи.

В версии Trusted Java 2.0 обеспечена обратная совместимость с продуктами КриптоАРМ и КриптоАРМ ГОСТ в плане создания и проверки ЭЦП и шифрования данных. Это, в частности, достигнуто за счет адаптации JCE провайдера для использования его в связке с библиотеками от BounceCastle: оригинальной библиотеки JCE-провайдера bcprov-jdk15-145.jar и пропатченной bcmail-библиотеки bcmail-jdk15-145-tj20-rbbb.jar, ссылки на которые можно получить в центре загрузки. В результате появилась возможность создавать и проверять заверяющие подписи.

С появлением нового проекта стандарта в версии Trusted Java 2.0 введена поддержка нового пространства имен при создании XML-подписей.

В версии Trusted Java 2.0 улучшена работа с SSL-сокетами, усовершенствованы механизмы аутентификации сервера и клиента, реализована поддержка шифросюиты (CipherSuite) 0x81 (TLS_GOSTR341001_WITH_28147_CNT_IMIT). Версия Trusted Java 1.5.3 поддерживает работу с КриптоПро CSP 3.6.

> **Внимание**
>
> Использование классов, реализующих работу по алгоритму ГОСТ Р 34.10-94, совместно с КриптоПро CSP 3.6 и выше не поддерживается.

Версия Trusted Java 1.5.2 включает в себя как совершенно новые, так и оптимизированные по сравнению с более ранними версиями, функциональные возможности: - обеспечена совместимость JSSE с веб-сервером Apache Tomcat 5.5;

- добавлены новые классы для обеспечения совместимости XMLSig с MS XML;
- оптимизирована подпись группы файлов на одном ключе;
- добавлена возможность получения списка ключевых контейнеров, подключенных к системе.

Состав продукта

В состав программного продукта входит:

- Провайдер JCE
- Провайдер JSSE
- Документация
  - Руководство администратора
  - Руководство разработчика
- Описание новинок версии
- Лицензионное соглашение
- Приложение "Удалить" (только для ОС Windows)
- Программа управления лицензиями (только для ОС Windows)

Функциональные возможности продукта

Библиотека Trusted Java реализует в Java-приложениях сертифицированные криптографические алгоритмы, предоставляемые криптопровайдером КриптоПро CSP от компании «Крипто-ПРО».

Функциональные возможности	Направления работы	Возможные операции
Поддержка криптографических алгоритмов		

Функциональные возможности	Направления работы	Возможные операции
	ГОСТ 28147-89 (шифрование) ГОСТ Р 34.11-94 (хеширование) ГОСТ Р 34.10-94 (ЭЦП,поддержка оставлена только для совместимости с предыдущими версиями)	Предназначен для обеспечения конфиденциальности информации и контроля ее целостности посредством шифрования имитозащиты Предназначены для авторизации и обеспечения юридической значимости электронных документов при обмене ими между пользователями. Это достигается благодаря использованию процедур: создания ЭЦП, проверки ЭЦП
Работа с сертификатами и запросами на сертификат	Операции с цифровыми сертификатами Операции с запросами на сертификат	создание сертификата установка сертификата в хранилище работа со списками отзванных сертификатов (COC) поддержка CMS поддержка TSP поддержка OCSP
Криптографические операции	Электронная цифровая подпись (ЭЦП) данных Шифрование	создание ЭЦП проверка корректности ЭЦП по сертификату создание ЭЦП в XML документах проверка ЭЦП в XML документах
Загрузка Java-апплетов	возможность загрузки Java-апплетов по ГОСТ TLS для Internet Explorer 6 (и выше) с установленным Sun Java Plugin 1.5 (и выше)	
Поддержка JCE и JSSE	провайдер JCE провайдер JSSE	поддержка ГОСТ алгоритмов в протоколе на стороне Сервера поддержка двухфакторной аутентификации по ГОСТ сертификатам на стороне Клиента поддержка прокси на стороне Клиента поддержка ГОСТ в TSP

Требования к программному обеспечению

Для установки и функционирования программного продукта Trusted Java необходимы следующие компоненты:

Операционная система

- Windows 2000 SP4 и выше (32|64 bit),
- RedHat Enterprise Linux 4|5 (32|64 bit),

Криптопровайдер КриптоПро CSP версии 4.0;

Java 2 (Sun JRE версии 1.6 и выше);

Дополнительное ПО

- для Windows: библиотеки Microsoft Visual C++ 2008 SP1;
- для unix-платформ: библиотеки GCC.

Открыть статью на сайте

Установка продукта

Для удобства в дальнейшем будет использоваться следующее обозначение для указания идентификатора платформы, на которой установлено СКЗИ КриптоПро CSP.

Установка СКЗИ КриптоПро CSP

Процедура установки криптопровайдера КриптоПро CSP проходит в 3 этапа:

- ознакомление с требованиями к системе
- установка дистрибутива СКЗИ КриптоПро CSP
- изменение списка устройств хранения ключевой информации

Установка дистрибутива СКЗИ КриптоПро CSP под Windows

Установка дистрибутива должна производиться пользователем, имеющим права администратора. Перед установкой дистрибутива ПО СКЗИ КриптоПро CSP удалите все ранее существующие версии устанавливаемого ПО. Если модуль криптографической поддержки не удален, новая версия не будет установлена.

Для удаления ранее установленного ПО СКЗИ КриптоПро CSP используйте пункты основного меню Windows Пуск → Панель управления → Установка и удаление программ.

Для установки ПО запустите на исполнение файл дистрибутива. Далее следуйте стандартным инструкциям установщика. После завершения установки дистрибутива перезагрузите компьютер.

Изменение списка устройств хранения ключевой информации под Windows

При инсталляции программного обеспечения по умолчанию устанавливаются все модули, обеспечивающие работу с различными поддерживаемыми устройствами хранения ключевой информации. Если для работы с ПО СКЗИ необходимы дополнительные типы устройств работы с ключевыми носителями, выберите режим изменения их состава. Для этого:

- Откройте панель управления компьютером, используя пункты меню Пуск →Панель управления.
- В окне панели управления выберите значок КриптоПро CSP.
- В панели настройки СКЗИ КриптоПро CSP выберите закладку «Оборудование» и, нажав кнопку «Настроить считыватели...», добавьте (или удалите) из списка те устройства, которые будут (или не будут) использованы в качестве считывателей ключевой информации.

Установка дистрибутива СКЗИ КриптоПро CSP под Linux

1. Установите криптопровайдер КриптоПро CSP 4.0 из дистрибутива.

   1.1. Для систем с поддержкой RPM-пакетов установите RPM-пакеты командой rpm -ivh <файл_пакета> в следующем порядке:

   - основные пакеты криптопровайдера (для варианта исполнения КС1): `lsb-cprocsp-base` `lsb-cprocsp-rdr` – базовые считыватели `lsb-cprocsp` – библиотеки криптопровайдера `lsb-cprocsp-capilite` – библиотеки CryptoAPI 2.0 Lite и утилиты
   - дополнительно может потребоваться установка вспомогательных пакетов: `curl` – библиотека для скачивания файлов по FTP, HTTP и некоторым другим протоколам `lsb-cprocsp-rdr-pcsc` – считыватели токенов и смарт-карт

2. Введите лицензию на КриптоПро CSP 4.0 с помощью команды `/opt/cprocsp/sbin/<arch>/cpconfig -license -set XXXXX-XXXXX-XXXXX-XXXXX-XXXXX`

Без установленной лицензии КриптоПро CSP 4.0 будет работать 3 месяца в полнофункциональном режиме. Приобрести бессрочную лицензию можно в магазине: https://cryptoarm.ru

Установка Trusted Java под Windows

Перед установкой Trusted Java на платформе семейства Microsoft Windows желательно, чтобы установка Java Runtime Environment уже была произведена, т.к. в ней будут отражены изменения в процессе установки Trusted Java . Их можно будет увидеть в окне инсталлятора. Для установки Trusted Java выполните команду, например, TrustedJava_x86_2.0.465.exe

Если текущая ОС поддерживает механизм контроля учетных записей пользователей (UAC), то потребуется разрешить запуск инсталлятора от имени администратора. Далее установка продолжается стандартным образом. После установки Trusted Java может потребоваться установить в системе Microsoft Visual C++ 2008 SP1 Redistributable Package, если такового в системе нет. Для этого в основном меню Пуск\Все Программы Digt\Trusted Java 2.0 выбрать соответствующий пункт и пройти по основным шагам его установки.

После установки может потребоваться из подкаталога jars каталога установки скопировать jar-файлы в каталог $JRE\lib\ext. В частности, для поддержки операций с XML-файлами это будут serializer.jar, xalan.jar, xmlsec.jar. Дополнительно могут понадобиться commons-logging.jar и junit.jar.

Установка Trusted Java под Linux

Предполагается, что перед установкой Trusted Java уже была проведена установка Java Runtime Environment, например, в каталог /usr/java/jre1.6.0_17. При дальнейшем рассмотрении будем считать, что значение этого каталога присвоено переменной окружения JRE. Данный продукт поставляется в виде следующих пакетов:

- TrustedJava---.zip (для Windows)
- TrustedJava---.tar.gz (для Unix-систем с КриптоПро CSP 3.6 и 3.6 R2)
- TrustedJava--R3--.tar.gz (для Unix-систем с КриптоПро CSP 3.6 R3 и выше)
- TrustedJava--R3--.tar.gz (для Unix-систем с КриптоПро CSP 4.0)
- trustedjava--..rpm (для Unix-систем)

Процесс установки производится с полномочиями пользователя root. Нужно распаковать в корень диска содержимое дистрибутива Trusted Java:

```
tar -xf ./TrustedJava_2.0_Linux.RHEL4.i386_r455.tgz -C /
```

В каталоге $JRE/lib/i386 требуется создать символическую ссылку на файл `"/opt/DIGT/Trusted Java 2.0/lib/<arch>/libdjcp20.so"`:

```
ln -s "/opt/DIGT/Trusted Java 2.0/lib/<arch>/libdjcp20.so" $JRE/lib/i386/libdjcp20.so
```

В каталоге $JRE/lib/ext требуется также создать символические ссылки на все jar-файлы в каталоге "/opt/DIGT/Trusted Java 2.0/jars":

```
ln -s "/opt/DIGT/Trusted Java 2.0/jars/commons-logging.jar" $JRE/lib/ext/commons-logging.jar
ln -s "/opt/DIGT/Trusted Java 2.0/jars/junit.jar" $JRE/lib/ext/junit.jar
ln -s "/opt/DIGT/Trusted Java 2.0/jars/trusted_java20.jar" $JRE/lib/ext/trusted_java20.jar
ln -s "/opt/DIGT/Trusted Java 2.0/jars/xalan.jar" $JRE/lib/ext/xalan.jar
ln -s "/opt/DIGT/Trusted Java 2.0/jars/xmlsec.jar" $JRE/lib/ext/xmlsec.jar
ln -s "/opt/DIGT/Trusted Java 2.0/jars/serializer.jar" $JRE/lib/ext/serializer.jar
```

Для деинсталляции продукта и возможности возврата первоначальных настроек нужно сохранить файл $JRE/lib/security/java.security в `$JRE/lib/security/java.security.tj`:

```
cp $JRE/lib/security/java.security $JRE/lib/security/java.security.tj
```

В файле `$JRE/lib/security/java.security` требуется

- Добавить строчку: `security.provider.N=com.digt.trusted.jce.provider.DIGTProvider` где N – первый свободный номер провайдера

- Установить параметру ssl.SocketFactory.provider значение `com.digt.trusted.jsse.provider.DigtSocketFactory`, изменив его или добавив строчку: `ssl.SocketFactory.provider=com.digt.trusted.jsse.provider.DigtSocketFactory`

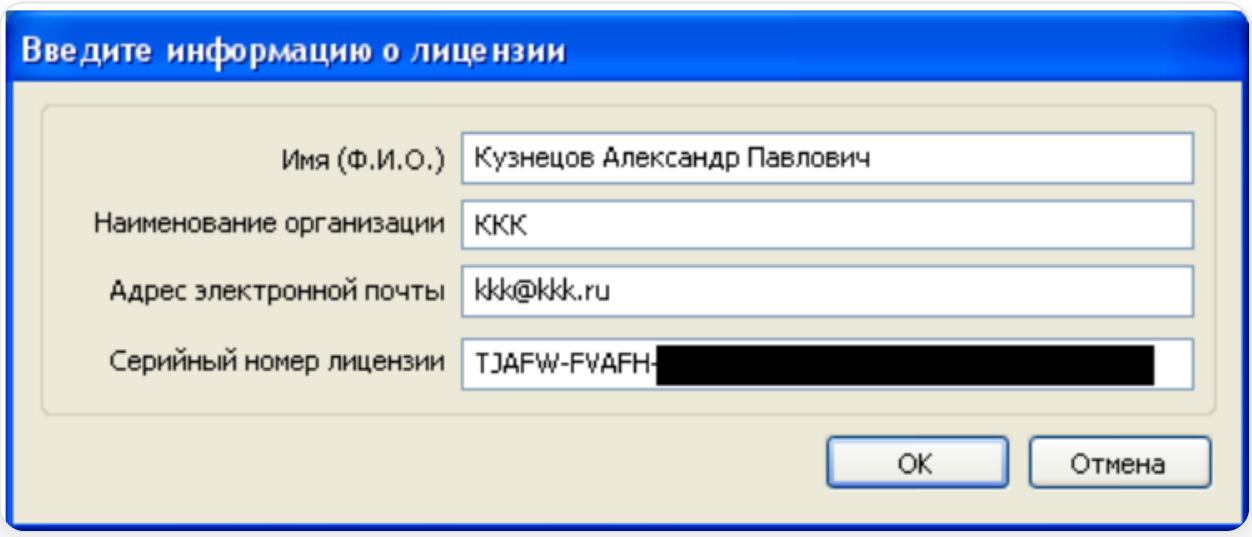
Установка лицензии на ПО

Последним шагом по установке TrustedJava остается ввести лицензию.

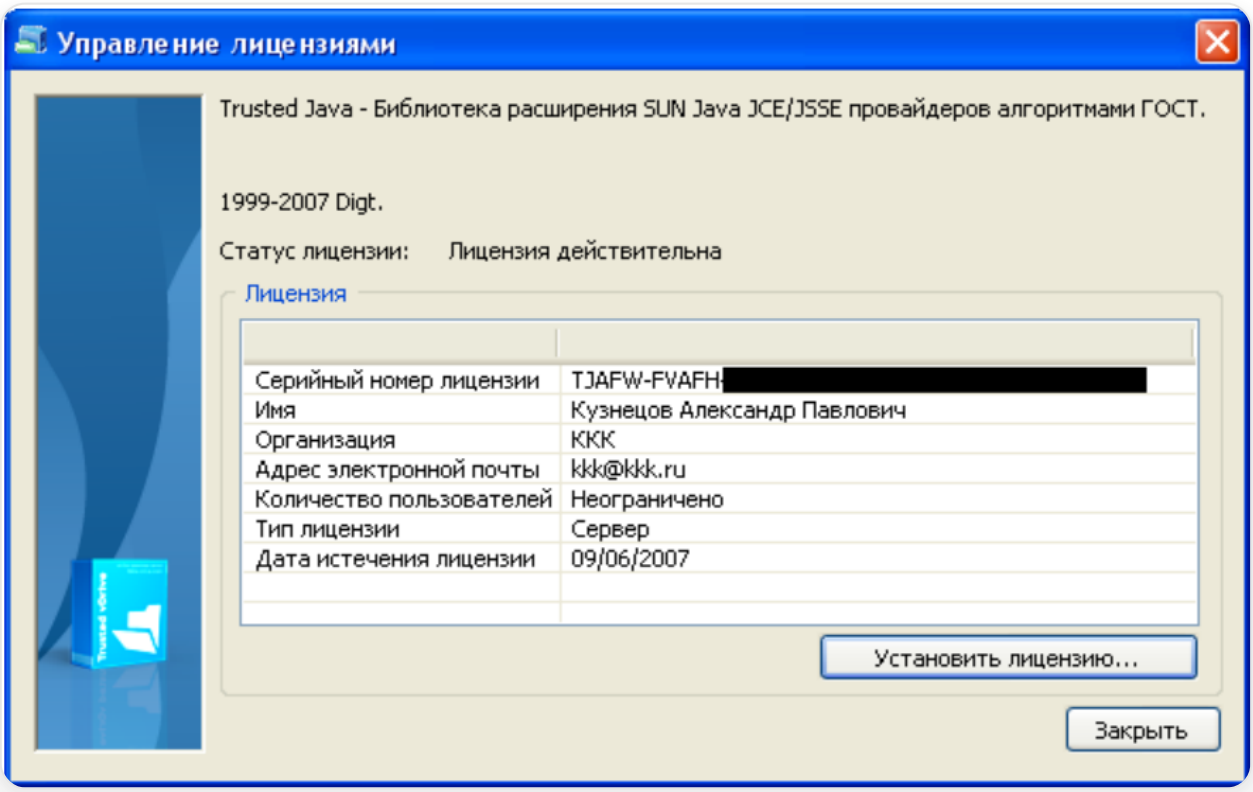
Установка лицензии для Windows

Процедура регистрации программного продукта различается в зависимости от операционной системы. Чтобы зарегистрировать программный продукт Trusted Java в ОС Windows:

- Откройте окно Управление лицензиями (Программы > Digt > Trusted Java 2.0 > Управление лицензиями).
- В этом окне нажмите на кнопку «Установить лицензию...».
- В открывшемся окне введите информацию о лицензии согласно выданным вам документам



При успешной регистрации ПО Trusted Java статус лицензии изменится на "Действительна" (см. в окне Управление лицензиями - > Статус лицензии):



В поле Лицензия отображается информация в соответствии с используемой лицензией на программный продукт Trusted Java:

- серийный номер лицензии
- имя администратора системы
- наименование организации, использующей данное ПО
- адрес электронной почты
- количество пользователей
- тип лицензии
- дата истечения срока действия лицензии на программный продукт

Установка лицензии для Linux

Чтобы зарегистрировать программный продукт Trusted Java в в Unix-системах, необходимо выполнить следующие действия: в дистрибутиве ПО существует файл license.lic, расположенный в каталоге /opt/DIGT/etc/Trusted/Java 2.0/license.lic. Откройте файл с помощью текстового редактора и впишите выданную вам лицензию в поле SerialNumber.

Проверка работоспособности ПО

Проверка работоспособности ПО под Linux

В каталоге установки /opt/DIGT/Trusted Java 2.0/tests находится архив тестов для проверки правильности установки и функционирования ПО.

Компиляцию и запуск тестов лучше провести под обычным пользователем, не root. Нужно распаковать архив

```
cd "/opt/DIGT/Trusted Java 2.0/tests"
unzip tests.zip
```

В файле build.sh нужно установить переменную JDK, значение которой должно указывать на каталог установки Java JDK, например, JDK=/usr/java/jdk1.6.0_07

Запускаем процесс компиляции

```
./build.sh
```

Далее перед запуском тестов нужно поменять переменную JRE в файле run.sh, указав в ее значении путь до каталога установки Java JRE JRE=/usr/java/jre1.6.0_17

Далее запустить тесты

```
./run.sh
```

Тестирование проходит достаточное количество времени и требует участия пользователя в виде нажатий на произвольные клавиши клавиатуры. Если тесты завершились примерно таким сообщением на консоли,

```
Test finished
Time: 406,109
OK (15 tests)
hKey pair is generated
Certificate is generated
X509Data element found
08.06.2010 17:07:43 org.apache.xml.security.signature.Reference verify
INFO: Verification successful for URI ""
The XML signature is valid
```

то результат тестирования можно считать успешным.

Открыть статью на сайте

Удаление продукта

Удаление ПО Trusted Java под Windows

Удалить ПО Trusted Java можно двумя способами:

- стандартными средствами ОС Windows
- через основное меню

Для удаления программы Trusted Java стандартными средствами ОС Windows:

- Откройте Пуск - > Панель управления, выберите опцию Установка и удаление программ
- В списке выберите запись Trusted Java 2.0 и нажмите на кнопку «Удалить» и подтвердите решение об удалении. Начнется процесс удаления ПО.

По завершении процесса библиотека Trusted Java будет удалена с компьютера и из списка элементов Установленные программы

Для удаления программы Trusted Java с помощью основного меню выберите приложение «Удаление Trusted Java 2.0»: Пуск > Программы > Digt > Trusted Java 2.0 > Удаление Trusted Java 2.0.

Удаление ПО Trusted Java под Linux

Процесс удаления производится с полномочиями пользователя root.

Нужно вернуть первоначальные настройки Java, записанные в файле `$JRE/lib/security/java.security`:

```
cp $JRE/lib/security/java.security.tj $JRE/lib/security/java.security
```

Если файл `$JRE/lib/security/java.security.tj` не существует, то в файле `$JRE/lib/security/java.security` нужно удалить строчку

```
security.provider.N=com.digt.trusted.jce.provider.DIGTProvider
```

где N – номер провайдера

Установить параметру ssl.SocketFactory.provider требуемое (первоначальное) значение, изменив его или удалив строчку

```
ssl.SocketFactory.provider=com.digt.trusted.jsse.provider.DigtSocketFactory
```

В каталоге $JRE/lib/ext требуется удалить символические ссылки на все jarфайлы в каталоге `"/opt/DIGT/Trusted Java 2.0/jars"`:

```
rm -f $JRE/lib/ext/trusted_java20.jar
rm -f $JRE/lib/ext/commons-logging.jar
rm -f $JRE/lib/ext/junit.jar
rm -f $JRE/lib/ext/serializer.jar
rm -f $JRE/lib/ext/xalan.jar
rm -f $JRE/lib/ext/xmlsec.jar
```

В каталоге $JRE/lib/ требуется удалить символическую ссылку на файл `"/opt/DIGT/Trusted Java 2.0/lib/<arch>/libdjcp20.so"`:

```
rm -f $JRE/lib/<arch>/libdjcp20.so
```

Остается только удалить распакованный дистрибутив продукта:

```
rm -Rf "/opt/DIGT/Trusted Java 2.0"
```

и каталог с лицензией на продукт:

```
rm -Rf "/opt/DIGT/etc/Trusted/Java 2.0"
```

Если установка продукта производилась из rpm-пакета, то деинсталлировать продукт можно штатным образом, используя команду

```
rpm -e trustedjava
```

Открыть статью на сайте

Настройка связки Trusted Java и Tomcat под Linux

Процесс установки и настройки описывается на примере Tomcat версий 5.5.28 и 6.0.20. Также проверялась совместимость с Tomcat 7.0.42. Номер версии, присутствующий в значениях каталогов, файлов, переменных окружения, можно успешно изменять, если не сказано иного. Отличия версии 6.0.20 описаны отдельно.

Требования к составу TLS-сертификатов

Для настройки SSL/TLS-шифрования для сервера потребуется X.509-сертификат, удовлетворяющий следующим требованиям:

- В имени владельца (т.е. в атрибуте «Субъект» / «Subject») сертификата элемент Common Name (CN) должен содержать конкретное доменное (DNS) имя сервера, на котором развертывается TLS-сервер, например, «webportal.yourcompany.ru», либо маску, например, «*.yourcompany.ru». Также, одно или несколько таких значений могут содержаться в расширении «Дополнительное имя субъекта» / «Subject Alternative Name».
- Ключевая пара должна обеспечивать возможность шифрования данных, что определяется наличием значений вариантов использования «Шифрование ключей, Шифрование данных» / «Key Encipherment, Data Encipherment» в расширении «Использование ключа» / «Key Usage» (KU) серверного сертификата.
- В расширении «Улучшенный ключ» / «Enhanced Key Usage» (EKU) должен содержаться объектный идентификатор 1.3.6.1.5.5.7.3.1, обозначающий вариант использования сертификата «Проверка подлинности сервера» / «Server Authentication».

Если необходимо, чтобы пользователи тоже аутентифицировались по сертификатам, то их сертификаты должны соответствовать следующим критериям:

- В расширении «Улучшенный ключ» / «Enhanced Key Usage» (EKU) должен присутствовать объектный идентификатор 1.3.6.1.5.5.7.3.2, обозначающий вариант использования сертификата «Проверка подлинности клиента» / «Client Authentication».
- Поддерживаются сертификаты без шифрования, то есть у которых в расширении «Использование ключа» / «Key Usage» (KU) указаны только «Цифровая подпись, Неотрекаемость» / «Digital Signature, NonRepudiation».

Установка Tomcat под Linux

С правами администратора создайте пользователя и одноименную группу tomcat, которым будет позволено запускать приложение Tomcat

```
groupadd tomcat
useradd -g tomcat tomcat
```

задайте пароль пользователю tomcat

```
passwd tomcat
```

распакуйте дистрибутив Tomcat в каталог /opt, например

```
tar -xf ./apache-tomcat-5.5.28.tar.gz -C /opt
```

присвойте соответствующие права на каталог с веб-сервером для полного доступа к его файлам настроек и логам пользователю tomcat, например:

```
chown -R tomcat:tomcat /opt/apache-tomcat-5.5.28
```

Запуск и остановка Tomcat под Linux

Войдите в систему пользователем tomcat, задайте переменные окружения CATALINA_HOME и JAVA_HOME, которые соответственно указывают на каталог установки Tomcat и JRE, например:

```
export CATALINA_HOME=/opt/apache-tomcat-5.5.28
export JAVA_HOME=/usr/java/jre1.6.0_17
```

Теперь запустите Tomcat:

```
$CATALINA_HOME/bin/startup.sh
```

Для остановки Tomcat выполните:

```
$CATALINA_HOME/bin/shutdown.sh
```

Настройка SSL в Tomcat под Linux

Настройка RSA шифрования в Tomcat под Linux

В данном разделе приводятся выдержки из документации на Tomcat со страниц http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html и http://tomcat.apache.org/tomcat-6.0-doc/ssi-howto.html. Под пользователем tomcat создайте новое хранилище ключей (certificate keystore), содержащее один самоподписанный сертификат, используя команду:

```
$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA
```

и укажите пароль для доступа к ключу, желательно, "changeit".

Дополнительно потребуется указать данные для сертификата – компанию, контактное лицо и т.д. При выполнении этой команды создается хранилище ключей в файле .keystore домашнего каталога пользователя.

Поддержка SSL в Tomcat реализуется двумя способами: JSSE реализация, предоставляемая частью Java Runtime (начиная с версии 1.4) APR реализация, которая включена по умолчанию в версии 5.5.28 и основана на OpenSSL

Дальнейшие действия по настройке приводятся для JSSE-реализации.

Для указания использования JSSE-реализации SSL, нужно раскомментировать узел с портом 8443 и схемой https в файле $CATALINA_HOME/conf/server.xml.

Для версии 5.5.28 раскомментировать строки:

```
 <Connector port="8443" maxHttpHeaderSize="8192"
 maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
 enableLookups="false" disableUploadTimeout="true"
 acceptCount="100" scheme="https" secure="true"
 clientAuth="false" sslProtocol="TLS" />
```

и изменить первую строку на:

```
<Connector protocol="org.apache.coyote.http11.Http11Protocol" port="8443"
maxHttpHeaderSize="8192"
```

Для версии 6.0.20 достаточно только раскомментировать строки:

```
 <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
 maxThreads="150" scheme="https" secure="true"
 clientAuth="false" sslProtocol="TLS" />
```

Опция «protocol» здесь уже установлена. После перезапуска Tomcat должна быть доступна страница https://servername:8443/

Настройка ГОСТ-шифрования в Tomcat под Linux

Для использования защищенного соединения по протоколу GOST TLS нужно указать тип протокола “GostTLS” в описании коннектора Tomcat. Для этого нужно в файле $CATALINA_HOME/conf/server.xml в узле с портом 8443 и схемой https заменить sslProtocol="TLS" на sslProtocol="GostTLS". Пример для версии 5.5.28:

```
<Connector protocol="org.apache.coyote.http11.Http11Protocol" port="8443"
maxHttpHeaderSize="8192"
 maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
 enableLookups="false" disableUploadTimeout="true"
 acceptCount="100" scheme="https" secure="true"
 clientAuth="false" sslProtocol="GostTLS" />
```

Пример для версии 6.0.20:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="GostTLS" />
```

Убедитесь, что при установке TrustedJava в файле `$JRE/lib/security/java.security` параметру `ssl.SocketFactory.provider` было присвоено значение `com.digt.trusted.jsse.provider.DigtSocketFactory`.

Установка серверного сертификата под Linux

Чтобы стартовать GOST TLS на ГОСТ сертификатах, необходимо на стороне сервера иметь ключевой контейнер и соответствующий ему сертификат в файле в формате "Base-64 encoded X.509":

- Сертификат должен иметь привязку к секретному ключу пользователя (т.е. установлен в хранилище "my" пользователя tomcat).
- Сертификаты УЦ, его подписавшие, должны находиться в хранилище "root" Unix платформы

Для выполнения этих требований необходимо выполнить следующие шаги: 1. Генерация запроса на сертификат (ГОСТ 34.10-2001) для веб-сервера 2. Получение сертификата УЦ и веб-сервера 3. Установка сертификата УЦ 4. Установка сертификата веб-сервера 5. Установка системных переменных окружения Java

Перед генерацией ключевого контейнера требуется определиться с его именем и местоположением. В данном примере описывается создание ключевого контейнера с именем «tlsserver» и расположенном на жестком диске (считыватель «HDIMAGE»). Имя ключевого контейнера должно быть уникальным, поэтому предварительно рекомендуется убедиться в отсутствии контейнера с выбранным именем. Список существующих ключевых контейнеров можно посмотреть командой: `/opt/cprocsp/bin/<arch>/csptestf -keyset -provtype 75 -enum_containers -verifycontext -fqcn`

Для генерации запроса на сертификат необходимо выполнить под учетной записью пользователя tomcat (от имени которого запускается сервер) следующую команду: `/opt/cprocsp/bin/<arch>/cryptcp -creatrqst -provtype 75 -ex -cont "\\\\.\\hdimage\\tlsserver" -dn " CN=web-portal.yourcompany.ru, O=My Company, C=RU, E=test@test.ru " -certusage "1.3.6.1.5.5.7.3.1" ~/tlssreq.csr`

Параметры -cont и -dn нужно изменить в соответствии с вашими данными.

Параметр -cont должен быть уникальным для каждого нового запроса, в примере он создаст ключевой контейнер в хранилище CSP на файловой системе, если вам нужно сгенерировать ключ на токене или дискете, измените его согласно руководству по КриптоПро CSP.

В процессе формирования ключа для инициализации датчика случайных чисел Вам может потребоваться нажать поочередно несколько клавиш на клавиатуре. После генерации закрытого ключа будет предложено защитить его паролем, который в дальнейшем можно менять командой: `/opt/cprocsp/bin/<arch>/csptestf -passwd -change <новый_пин_код> -provtype 75 -container <имя_контейнера>`

ПОЛУЧЕНИЕ СЕРТИФИКАТОВ УЦ И ВЕБ-СЕРВЕРА

Сгенерированный файл запроса ~/tlssreq.csr следует обработать в Удостоверяющем центре. При тестировании можно использовать тестовый УЦ компании КриптоПро: http://www.cryptopro.ru/certsrv/certrqxt.asp. На данной странице в поле «Base-64-шифрованный запрос сертификата (CMC или PKCS #10 или PKCS #7)» нужно вставить содержимое файла запроса и продолжить работу, нажав на кнопку «Выдать >». Затем следует подтвердить передачу данных по незашифрованному каналу, в следующем окне обязательно выбрать «Base64-шифрование» и выполнить пункт «Загрузить сертификат». Полученный в УЦ сертификат веб-сервера сохраните на сервере в файл ~/servergost.cer.

Для получения сертификата УЦ нужно зайти на страницу http://www.cryptopro.ru/certsrv, выбрать пункт «Получить сертификат Удостоверяющего Центра или действующий список отозванных сертификатов» и продолжить работу, нажав на кнопку «Дальше >> >». В следующем окне следует выбрать «Base64-шифрование» и выполнить пункт «Загрузка сертификата ЦС». Сохраните сертификат УЦ на сервере в файл ~/caservergost.cer.

УСТАНОВКА СЕРТИФИКАТА УЦ

Сертификат корневого Удостоверяющего центра должен быть установлен в хранилище корневых ("root") сертификатов. Для установки корневого сертификата УЦ в машинное хранилище корневых сертификатов получите привилегии супер-пользователя и выполните следующую команду: `/opt/cprocsp/bin/<arch>/certmgr -inst -store mRoot -file ~tomcat/caservergost.cer -cert`

Если команда установки сертификата выполнилась с кодом ошибки 0x00000000, значит, сертификат УЦ успешно установлен. В случае если серверный сертификат издан промежуточным УЦ, то сертификат последнего должен быть установлен в хранилище сертификатов промежуточных УЦ ("ca"), а корневой, которым подписан промежуточный, в хранилище корневых сертификатов ("root"). Установка сертификата корневого УЦ описана выше.

Установка промежуточных сертификатов в машинное хранилище должна выполняться с привилегиями супер-пользователя командой `/opt/cprocsp/bin/<arch>/certmgr -inst -store mCa -file ~tomcat/intermediategost.cer -cert`

УСТАНОВКА СЕРТИФИКАТА ВЕБ-СЕРВЕРА

Для установки в хранилище my пользователя tomcat сертификата веб-сервера, полученного на предыдущем этапе (обязательно в формате "Base64"), необходимо от его имени выполнить команду `/opt/cprocsp/bin/<arch>/cryptcp -instcert -provtype 75 ~/servergost.cer` и в появившемся пронумерованном списке контейнеров указать номер соответствующего контейнера и пароль на данный контейнер.

В случае, если предыдущая команда завершится с ошибкой, можно попробовать установить серверный сертификат другим способом: `/opt/cprocsp/bin/<arch>/certmgr -inst -store uMy -file ~/servergost.cer -cont "\\\\.\\HDIMAGE\\tlsserver"`

Если одна из предыдущих команд установки сертификата выполнилась с кодом ошибки 0x00000000 или сообщением «Certificate is installed.», значит, серверный сертификат установлен в личное хранилище текущего пользователя со ссылкой на закрытый ключ. Проверить наличие установленного серверного сертификата в личном ("my") хранилище сертификатов пользователя tomcat можно, выполнив следующую команду: `/opt/cprocsp/bin/<arch>/certmgr -list -store uMy -cert`

В выведенном на экран списке в блоке соответствующего сертификата должна присутствовать строчка: PrivateKey Link: Yes. Container: <имя_контейнера>

УСТАНОВКА СИСТЕМНЫХ ПЕРЕМЕННЫХ ОКРУЖЕНИЯ JAVA

Скопируйте сертификат веб-сервера в каталог /opt/apache-tomcat-5.5.28/conf:

```
cp ~/servergost.cer /opt/apache-tomcat-5.5.28/conf
```

Для указания серверного сертификата и пароля к ключу соответствующего контейнера приватного ключа используются следующие параметры системного окружения Java:

```
com.digt.trusted.jsse.server.certFile
com.digt.trusted.jsse.server.keyPasswd
```

В связи с этим нужно добавить в начале файла $CATALINA_HOME/bin/startup.sh следующую строку

```
export JAVA_OPTS="$JAVA_OPTS -Dcom.digt.trusted.jsse.server.certFile=/opt/apache-tomcat5.5.28/
conf/servergost.cer -Dcom.digt.trusted.jsse.server.keyPasswd=*******"
```

Замените «*» в пароле на конкретное его значение.

Открыть статью на сайте

Java Bridge: Интеграция Trusted Java и PHP

Настройка доступа к Trusted Java из PHP под APACHE

Вопрос установки http-сервера APACHE, PHP и модуля PHP для APACHE здесь не рассматривается. Установка Trusted Java и JRE описана выше. Далее описывается процесс настройки доступа из модуля PHP для APACHE к Java и, как следствие, к Trusted Java.

Сначала требуется получить файлы со страницы PHP/Java Bridge (требующие Java 1.6 и выше) и разместить их в каталоге, например, /opt/JavaBridge:

```
/opt/JavaBridge/java/Java.inc
/opt/JavaBridge/ext/JavaBridge.jar
/opt/JavaBridge/ext/php-script.jar
/opt/JavaBridge/ext/php-servlet.jar
```

Создайте в каталоге /opt/JavaBridge файл PHPJavaBridge_start.sh:

```
#!/bin/sh
java -Dphp.java.bridge.daemon="true" -jar /opt/JavaBridge/ext/JavaBridge.jar SERVLET_LOCAL:
8080 3
JavaBridge.log
```

Перед его запуском в режиме «демона» необходимо убедиться, что при запуске java запускается установленная ранее версия JRE. Настроить использование требуемой JRE с использованием alternatives возможно, например, следующим образом:

```
alternatives --install /usr/bin/java java /usr/java/jre1.6.0_17/bin/java 10000
```

Теперь можно выполнить файл `/opt/JavaBridge/ PHPJavaBridge_start.sh`.

Откройте www-каталог http-сервера, например, /var/www/html и создайте в нем символическую ссылку JavaBridge на каталог /opt/JavaBridge. Положите в него тестовый файл SignAndVerify.php из архива с тестами, поставляемыми в составе дистрибутива Trusted Java. Запустите http-сервер. Проверить работу связки Trusted Java + PHP Apache + PHP Java Bridge можно обратившись из Internet Explorer по ссылке http://servername/SignAndVerify.php. В данном примере реализовано формирование ЭЦП из браузера с использованием ПО КриптоАРМ, а ее проверка на стороне сервера – с использованием Trusted Java.

Настройка доступа к Trusted Java из PHP под Tomcat

Требуется получить файл JavaBridge.war, который появляется после установки rpm-пакета, для дальнейшего его включения в сервер Tomcat6. Для создания rpm-пакета PHP Java Bridge необходимо наличие следующих компонент:

- Java JDK версии 1.4.2 или выше (рекомендуется версия 6)
- PhpDocumentor версии 1.4.2 или выше
- Ant версии 1.7.1 или выше
- Apache HTTP Server

При использовании SELinux:

- selinux-policy – конфигуратор политик,
- selinux-policy-devel – пакет разработки,
- policycoreutils – утилиты для работы с политиками,
- checkpolicy – компилятор политик,
- coreutils – набор основных утилит GNU.

Для установки rpm-пакета PHP Java Bridge потребуются следующие компоненты:

- PHP версии 5.1.2 или выше (рекомендуется версия 5.3 или выше)
- Tomcat версии 6.x

Далее потребуется настроить установку пакетов, например, через jpackageutil (Jpackage Project), подключив, например, к yum репозитарий jpackage50.repo.

Таким образом можно будет установить Ant, Tomcat6. Rpm-пакет PhpDocumentor можно получить, например, с сайта RPM pbone.net. Там же могут быть получены связанные с ним пакеты. Скачайте архив php-java-bridge_6.1.2.1.tar.gz, разместите в каталоге пользователя root, например, /root/soft. Для сборки rpm-пакета выполните команду

```
rpmbuild -tb php-java-bridge_6.1.2.1.tar.gz
```

Если появляется сообщение

```
ошибка: синтаксическая ошибка в выражении
ошибка: /usr/src/redhat/SPECS/php-java-bridge.spec:56: parseExpressionBoolean код возврата: -1
ошибка: Пакет не имеет %description: php-java-bridge
```

то создайте и исполните файл /root/soft/deploy.sh:

```
#!/bin/sh
cd /root/soft
tar -zxf php-java-bridge_6.1.2.1.tar.gz *.spec
mv php-java-bridge-6.1.2.1/php-java-bridge.spec .
rm -R php-java-bridge-6.1.2.1
cp php-java-bridge_6.1.2.1.tar.gz /usr/src/redhat/SOURCES/
```

Далее в полученном файле /root/soft/php-java-bridge.spec закомментируйте строки 56-57:

```
…
#%if (%{PHP_MINOR_VERSION} == 2 && %{PHP_RELEASE_VERSION} > 6) || (%{PHP_MINOR_VERSION} >
2)
#%endif
…
```

и выполните команду:

```
rpmbuild -bb /root/soft/php-java-bridge.spec
```

Полученные rpm-файлы можно забрать из каталога /usr/src/redhat/RPMS/i386.

Установите полученный пакет PHP Java Bridge, выполнив команду

```
rpm -i /usr/src/redhat/RPMS/i386/php-java-bridge-6.1.2.1-1.i386.rpm
```

В приведенной команде может использоваться опция --nodeps для случая, когда Tomcat6 был установлен, но не через механизм rpm-пакетов. В этом случае требуется вручную довести установку PHP Java Bridge для связки с Tomcat6. Скопируйте файл JavaBridge.war в подкаталог webapps каталога установки Tomcat6:

```
cp /var/lib/tomcat6/webapps/JavaBridge.war /opt/apache-tomcat-6.0.20/webapps
```

Перезапустите Tomcat6:

```
service tomcat6 restart
```

Теперь должна быть доступна страница с тестами

Открыть статью на сайте

Сервера приложений: интеграция Trusted Java и TrustedTLS

В предлагаемом ниже материале рассматриваются методики построения защищенного канала до серверов приложений (Tomcat, IBM Websphere, Oracle Weblogic) на базе продукта TrustedTLS (веб-сервера Apache версии 2.2) для обеспечения конфиденциальности информации и аутентификации пользователей в приложениях, разворачиваемых на этих серверах. (Методики были проверены на операционной системе RedHat Enterprise Linux/ CentOS 5.3 x86.)

В частности, рассматриваются вопросы обеспечения взаимодействия в цепочке <Клиент> - и - <Сервер приложений>, а также доставки сертификатов пользователей (и Apacheсервера) до сервера приложений.

Продукт Trusted Java также может быть интегрирован в составе рассматриваемых решений как мост для доступа приложений на Java к криптографическим операциям при использовании следующих связок:

```
Apache + Trusted TLS 2.2 + CSP 3.6/3.6 R2 + Tomcat 5.5/6.0/7.0 + Trusted Java
Apache + Trusted TLS 2.2 + CSP 3.6/3.6 R2 + IBM Websphere 6.1/7.0 + Trusted Java
Apache + Trusted TLS 2.2 + CSP 3.6/3.6 R2 + Oracle WebLogic Server 10.3.2 + Trusted Java
```

Передача сертификатов от Apache-сервера серверам приложений

Для использования сертификатов клиента (и сервера) на стороне развернутого приложения предлагается использовать на стороне Apache-сервера возможность модуля mod_headers вставлять в заголовок запроса переменную с содержимым из SSL-переменных.

Предполагается, что на Apache-сервере уже настроен модуль `mod_digt_tls.so` (продукт TrustedTLS) согласно прилагаемым к нему инструкциям.

В конфигурационном файле httpd.conf добавим строку загрузки модуля mod_headers LoadModule headers_module modules/mod_headers.so и в конфигурационном файле ssl.conf в секции прописываем строки

```
<IfModule mod_headers.c>
RequestHeader set Forwarded-SSL-CLIENT-CERT "%{SSL_CLIENT_CERT}s"
RequestHeader set Forwarded-SSL-SERVER-CERT "%{SSL_SERVER_CERT}s"
</IfModule>
```

Пересылка запросов серверу приложений от Apache-сервера

Пересылку запросов на сервер приложений можно организовать с использованием

- Apache-модуля mod_proxy, Специализированного WebServer Plug-ins
- Apache-модуль Mod_proxy

Процесс интеграции

В конфигурационном файле httpd.conf добавим строку загрузки модуля mod_proxy.so

```
LoadModule proxy_module modules/mod_proxy.so
```

и в конфигурационном файле `ssl.conf` в секции прописываем строки

```
<Location />
    <IfModule mod_proxy.c>
        ProxyPass http://AS-host:AS-port/
        ProxyPassReverse http://AS-host:AS-port/
    </IfModule>
</Location>
```

As-host и AS-port – DNS-имя и порт хоста, на котором принимает запросы сервер приложений.

Mod_proxy и ApacheTomcat 5/6/7

Для интеграции с ApacheTomcat 5/6/7 в качестве порта AS-port нужно использовать значение 8080. Например,

```
<Location />
    <IfModule mod_proxy.c>
        ProxyPass http://localhost:8080/
        ProxyPassReverse http://localhost:8080/
    </IfModule>
</Location>
```

Mod_proxy и IBM Websphere 6.1/7.0

Для интеграции с IBM Websphere 6.1/7.0 в качестве порта AS-port нужно использовать значение 9080. Например,

```
<Location />
    <IfModule mod_proxy.c>
        ProxyPass http://localhost:9080/
        ProxyPassReverse http://localhost:9080/
    </IfModule>
</Location>
```

Mod_proxy и Oracle Weblogic 10.3.2

Для интеграции с Oracle Weblogic 10.3.2 в качестве порта AS-port нужно использовать значение 7001. Например,

```
<Location />
    <IfModule mod_proxy.c>
        ProxyPass http://localhost:7001/
        ProxyPassReverse http://localhost:7001/
    </IfModule>
</Location>
```

Apache-Tomcat mod_jk connector

Для построения связки между Apache и Tomcat серверами будем использовать Apache Tomcat Connector.

Из хранилища выкачиваем для Apache 2.2 (например, под Linux) `mod_jk1.2.28-httpd-2.2.X.so`. Копируем его под именем `mod_jk.so` в `APACHE_HOME/modules`.

Конфигурирование сервера Apache

Проверяем, что в файле `APACHE_HOME/conf/httpd.conf` задана директива `Include conf.d/*.conf`

Создаем конфигурационный файл `APACHE_HOME/conf.d/mod_jk.conf` со следующим содержимым:

```
LoadModule jk_module modules/mod_jk.so
JkWorkersFile "conf.d/workers.properties"
# Where to put jk shared memory
JkShmFile "logs/mod_jk.shm"
JkLogFile "logs/mod_jk.log"
JkLogLevel info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]”
```

В соответствие директиве JkWorkersFile создаем далее конфигурационный файл `APACHE_HOME/conf.d/workers.properties` со следующим содержимым:

```
[channel.socket:localhost:8009]
port=8009
host=localhost
worker=ajp13:localhost:8009
```

Он описывает параметры AJP-соединения: идентификатор соединения с именем хоста, используемое значение порта и тип соединения. В файле APACHE_HOME/conf.d/ssl.conf настраиваем в секции виртуального хоста:

```
<VirtualHost _default:443>
…
 JkMount /* ajp13
 JkLogLevel info
…
</VirtualHost>
```

Перезапускаем сервер Apache.

Конфигурирование сервера Tomcat

Согласно содержимому конфигурационного файла APACHE_HOME/conf.d/workers.properties в конфигурационном файле `TOMCAT_HOME/conf/server.xml` сервера Tomcat проверяем наличие строк

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" enableLookups="false" redirectPort="8443"
protocol="AJP/1.3" />
```

Перезапускаем сервер Tomcat и заходим на ресурсы сервера TOMCAT по защищенному каналу на порту 443.

Oracle WebLogic WebServer Plug-Ins

Oracle WebLogic WebServer Plug-Ins входит в поставку Oracle WebLogic 10.3.2. После установки Oracle WebLogic 1. Взять из каталога WL_HOME/server/plugin модуль mod_wl_22.so (или mod_wl128_22.so) из подкаталога, соответствующего платформе, на которой установлен web-сервер Apache (например linux/i686). 2. Скопировать его в APACHE_HOME/modules. 3. В конфигурационный файл APACHE httpd.conf прописать загрузку этого модуля: LoadModule weblogic_module modules/mod_wl_22.so 4. Установить проксирование на базе путей (можно указать в секции

```
<Virtualhost _default_:4433>):
<Location />
 SetHandler weblogic-handler
 WLLogFile /tmp/wl_root_log.log
</Location>
```

5. Установить глобальные параметры:

```
<IfModule mod_weblogic.c>
 WebLogicHost localhost
 WebLogicPort 7001
 Debug OFF
 WLLogFile /tmp/wl_global_proxy.log
 #WLTempDir "/tmp/wl"
 DebugConfigInfo On
 KeepAliveEnabled ON
 KeepAliveSecs 15
</IfModule>
```

После проделанных действий можно использовать доступ к серверу Weblogic по порту 4433.

IBM WebSphere 7.0 WebServer Plug-Ins

Далее описывается поцесс настройки Apache 2.2 WebServer Plug-Ins для WebSphere 7.0. Установка плагина производится штатным образом согласно документации с дополнительного установочного диска.

После установки плагина нужно проверить в файле httpd.conf наличие следующих строк для Linux или Solaris x32 LoadModule was_ap22_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap22_http.so WebSpherePluginConfig /opt/IBM/WebSphere/Plugins/config/webserver_ap22/plugin-cfg.xml

айл plugin-cfg.xml генерируется в процессе установки или через консоль управления сервером (https://servername:9043/ibm/console/logon.jsp) после внесения различных изменений в конфигурацию сервера приложений. По умолчанию в результате этой настройки по портам 80 (для http) и 443 (для https) запросы будут транслироваться от Apache-сервера на сервер приложений.

Для конфигурирования https-протокола на нестандартный порт, например, 4433 требуется проделать следующие действия, которые описаны для варианта размещения сервера приложений и apache-сервера на одном хосте.

- Зарегистрироваться в консоли управления сервером.
- В пункте «Среда» выбрать «Виртуальные хосты».
- В списке виртуальных хостов выбрать, например, «default_host».
- В «Дополнительных свойствах» выбрать «Псевдонимы хоста».
- Выбрав пункт «Создать», в поле «Имя хоста» внести DNS-имя хоста, в поле «Порт» указать значение 4433 и нажать «Ок».
- Далее требуется сохранить конфигурацию.
- После определения виртуального хоста требуется в пункте «Среда» выбрать «Обновить глобальную конфигурацию модуля Webсервера» и затем нажать кнопку «Ок».
- Далее в пункте «Среда» выбрать «Серверы» - «Типы серверов» - «Web-серверы».
- В поле «Выбрать» напротив требуемого сервера выставить галочку.
- Последовательно выбрать пункты «Сгенерировать модуль» и «Распространить модуль».

Перезапустить сервер приложений. В конфигурационном файле ssl.conf apache-сервера нужно проверить наличие строк

```
…
Listen 4433
…
<Virtualhost [DNS-имя хоста|_default_]:4433>
…
</Virtualhost>
```

Перезапустить apache-сервер. После проделанных действий будут доступны развернутые приложения на виртуальном сервере default_host по защищенному каналу на порту 4433.

Открыть статью на сайте

Часто задаваемые вопросы

при встраивании Java-библиотеки:

Описание ошибки	Рекомендации по устранению
Java сообщает, что не найден провайдер DIGT	Добавьте в начало кода строку: Security.addProvider(new DIGTProvider())
Ошибка: java.lang.ClassNotFoundException: com.digt.trusted.jce.provider.DIGT Provider	Поместите trusted_java20.jar в каталог /lib/ext.
Ошибка: java.lang.UnsatisfiedLinkError: no djcpNNN in java.library.path	Java не может найти библиотеку. Поместите ее по одному из путей переменных окружения %PATH% либо укажите путь к ней с помощью параметра java машины -Djava.library.path.
Ошибка java.lang.UnsatisfiedLinkError: libdjcp20.so: undefined symbol: CertFreeCertificateContext	Перед запуском JVM добавьте в переменную окружения LD_PRELOAD путь к библиотеке libcapi20.so.3 из КриптоПро CSP, например: export LD_PRELOAD="/opt/cprocsp/lib//libcapi20.so.3.0.0:$LD_PRELOAD"

при использовании Trusted Java с Tomcat

Описание ошибки	Рекомендации по устранению
При запуске Tomcat в его лог-файле catalina.out присутствуют строки SEVERE: Failed to load keystore type JKS with path C:\Documents and Settings\/.keystore due to C:\Documents and Settings\.keystore (The system cannot find the file specified) java.io.FileNotFoundException: C:\Documents and Settings\.keystore (The system cannot find the file specified) или (для локализованной ОС): SEVERE: Failed to load keystore type JKS with path C:\Users\/.keystore due to C:\Users\.keystore (Не удается найти указанный файл) java.io.FileNotFoundException: C:\Users\.keystore (Не удается найти указанный файл)	Сформируйте keystore с RSAсертификатом в соответствии с описанием в разделе «Настройка RSA шифрования в Tomcat под Linux».
При запуске Tomcat в его лог-файле catalina.out присутствует строка java.io.IOException: GostTLS SSLContext not available	Пропишите DIGTProvider в java.security, т.е. добавьте строку вида: security.provider.NN=com.digt.trusted.jce.provider.DIGTProvider Также убедитесь, что в каталоге /lib/ext присутствует файл trusted_java20.jar.

Описание ошибки	Рекомендации по устранению
Не открывается страница https://servername:8443 и в лог-файле сервера catalina.out присутствуют строки INFO: Initializing Coyote HTTP/1.1 on http-8080 11.06.2010 12:38:43 com.digt.trusted.jsse.provider.DigtSSLContext getDefaultContext SEVERE: Error setting certificate. 11.06.2010 12:38:43 org.apache.coyote.http11. Http11BaseProtocol init SEVERE: Error initializing endpoint	Проверьте правильность указания пути к файлу сертификата веб-сервера в переменной окружения JAVA_OPTS. Внимание! При объявлении JAVA_OPTS под Windows параметры certFile и keyPasswd нужно обрамлять кавычками по отдельности. Если сертификат сохранен в кодировке Base64, то удостоверьтесь в наличии Base64-заголовков (-----BEGIN CERTIFICATE----- и -----END CERTIFICATE-----). Если сертификат сервера квалифицированный и установленная сборка Trusted Java ниже 490, то обновите Trusted Java.
Не открывается страница https://servername:8443 и в лог-файле сервера catalina.out присутствуют строки INFO: Server startup in 633 ms Password:Password: 11.06.2010 4:29:33 org.apache.tomcat.util.net.PoolTcpEndpoint acceptSocket SEVERE: Endpoint ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=84 4 3] ignored exception: java.io.IOException: error: 1408B06B:SSL routines:SSL3_GET_CLIENT_KEY_EXCHANGE:bad decompression	Проверьте правильность указания пароля к ключевому контейнеру в переменной окружения JAVA_OPTS, а также наличие обрамляющих кавычек «...» при ее описании.
Не открывается страница https://servername:8443 и в лог-файле сервера catalina.out присутствуют строки INFO: Server startup in 564 ms 11.06.2010 12:32:51 org.apache.tomcat.util.net.PoolTcpEndpoint acceptSocket SEVERE: Endpoint ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=84 43] ignored exception: java.io.IOException: error: 1409A044:SSL routines:SSL3_SEND_SERVER_CERTIFICATE:internal error	Проверьте правильность указания пути к файлу сертификата веб-сервера в переменной окружения JAVA_OPTS, а также наличие обрамляющих кавычек «...» при ее описании.
Не открывается страница https://servername:8443 и в лог-файле сервера catalina.out присутствуют строки java.lang.Exception: Connector attribute SSLCertificateFile must be defined when using SSL with APR или org.apache.tomcat.util.net.NioEndpoint	
setSocketOptions SEVERE: java.lang.NullPointerException	В Trusted Java не поддерживаются NIO-сокеты. Замените в конфигурации

Описание ошибки	Рекомендации по устранению
значение параметра protocol с «HTTP/1.1» или «org.apache.coyote.http11.Http11 NioProtocol» на «org.apache.coyote.http11.Http11 Protocol».	

не нашли описания своей проблемы?

Проверьте, не истек ли период использования Крипто Про CSP. В противном случае обратитесь в техническую поддержку (см. контакты ниже) и опишите сложившуюся ситуацию с примерами появившихся ошибок.

Техническая поддержка

По вопросам технической поддержки ПО Trusted Java обращайтесь:

- по электронной почте: support@trusted.ru
- по телефону: 8 (8362) 33-70-50
- по адресу: 424033, Россия, РМЭ, г. Йошкар-Ола, ул. Петрова, д.1, а/я 67.

Систематическое техническое сопровождение организаций по вопросам работы с ПО Trusted Java осуществляется при условии оплаты стоимости годового пакета технических услуг. В течение этого срока разработчик бесплатно поставляет новые реализации продукта в рамках приобретенной версии. Новые версии продукта поставляются в соответствии с прайс-листом.

О компании-разработчике

Компания «Цифровые технологии» – российский разработчик и поставщик программного обеспечения в области защиты информации, систем электронного документооборота и хранения данных. Основной сферой деятельности компании является разработка, внедрение и поддержка криптографических продуктов и решений для государственных и коммерческих структур.

- Телефон: (8362) 33-70-50
- Сайт: http://www.trusted.ru
- E-mail: info@trusted.ru

Открыть статью на сайте

Программные интерфейсы

Новинки версии

В сборке r507 доработано чтение сертификатов из хранилища. Ранее не возвращались сертификаты с алгоритмом публичного ключа ГОСТ 2012 (256 - 512). Использование:

```
CryptoProCSPCertStore storeCsp = new CryptoProCSPCertStore();
Collection certificates = storeCsp.getInstance("CurrentUser/ROOT",
null).getAllCertificates();
```

В сборке r506 добавлена возможность проверки подписи с отключением/включением сортировки подписанных атрибутов (по умолчанию они не сортируются. Использование:

```
SignerInformation si;
...
si.setNeedSortSignedAttributes(true);
si.verify(cert, PROV_NAME))
```

В сборке r505 добавлена поддержка алгоритмов ГОСТ 2012 для XML.

В версии Trusted Java 2.0 (сборка r504 и выше) добавлена поддержка работы с ключами и алгоритмами ГОСТ Р 34.11-2012/ГОСТ Р 34.10-2012 (только с КриптоПро CSP 4.0).

В версии Trusted Java 2.0 изменилась система лицензирования. Теперь полученный лицензионный ключ действителен только для одной платформы ОС и не подходит к другой. Поэтому следует пользователям, переходящим на новую версию, обновить или приобрести под другую платформу лицензионные ключи.

В версии Trusted Java 2.0 обеспечена обратная совместимость с продуктом КриптоАРМ в плане создания и проверки ЭЦП и шифрования данных. Это, в частности, достигнуто за счет адаптации JCE провайдера для использования его в связке с библиотеками от BounceCastle: оригинальной библиотеки JCEпровайдера bcprov-jdk15-145.jar и пропатченной bcmail-библиотеки bcmailjdk15-145-tj20-rbbb.jar, ссылки на которые можно получить в центре загрузки. В результате появилась возможность создавать и проверять заверяющие подписи.

В версии Trusted Java 2.0 улучшена работа с SSL-сокетами, усовершенствованы механизмы аутентификации сервера и клиента, реализована поддержка шифросюиты (CipherSuite) 0x81 (TLS_GOSTR341001_WITH_28147_CNT_IMIT).

Версия Trusted Java 1.5.3 поддерживает работу с КриптоПро CSP 3.6.

> **Внимание**
>
> Использование классов, реализующих работу по алгоритму ГОСТ Р 34.10-94, совместно с КриптоПро CSP 3.6 и выше не поддерживается.

Версия Trusted Java 1.5.2 включает в себя как совершенно новые, так и оптимизированные по сравнению с более ранними версиями, функциональные возможности:

- обеспечена совместимость JSSE с веб-сервером Apache Tomcat 5.5;
- добавлены новые классы для обеспечения совместимости XMLSig с MS XML;
- оптимизирована подпись группы файлов на одном ключе;
- добавлена возможность получения списка ключевых контейнеров, подключенных к системе.

Объекты управления

Объект	Описание
Шифрование и расшифрование	Шифрование – это процесс выбора данных (текст-оригинал) и короткой последовательности символов (ключ), и вывода данных (шифрованный текст), являющихся бессмысленными для тех, кто не знает код. Расшифрование – это обратный процесс, то есть выбор шифрованного текста и кодовой последовательности, и вывод текстаоригинала.
Шифрование с использованием пароля (PBE)	Шифрование с использованием пароля выводит ключ шифрования из пароля. Для того чтобы переход от пароля к ключу отнимал как можно больше времени при попытке несанкционированного доступа, в процессе шифрования многие элементы смешиваются, создавая так называемую «соль».
Шифр	Шифрование и расшифрование осуществляется с использованием шифров. Шифр – это объект, способный выполнять шифрование и расшифрование в соответствии со схемой шифрования (алгоритмом).
Согласование ключа	Согласование ключа – это протокол, при помощи которого 2 или более пользователя могут установить одни криптографические ключи, не прибегая к обмену какой-либо секретной информацией.

Объект	Описание
Код аутентификации сообщений	Код аутентификации сообщений (MAC), на основе закрытого ключа, создает способ проверки целостности информации, переданной или сохраненной на ненадежном носителе. Обычно такие коды используются двумя пользователями, обладающими закрытым ключом, для того, чтобы проверить цельность информации, передаваемой между ними. Код аутентификации сообщений, который основан на криптографических хэшфункциях, называют HMAC. HMAC может использоваться с любыми криптографическими хэш-функциями, например, MD5 или SHA1, в сочетании с закрытым совместным ключом. HMAC подробно описан в стандарте RFC 2104.

JCE и ее особенности

JCE создает структуру и реализацию шифрования, образования ключей и их согласования, а также алгоритмов аутентификации сообщений (MAC). Шифрование базируется на использовании симметричных, асимметричных, блочных и поточных шрифтов.

Программное обеспечение также поддерживает передачу защищенных потоков и закрытых объектов.

JCE основана на тех же структурных принципах, что и другие программы JCA: независимость реализаций и, если это возможно, независимость алгоритмов.

Данная программа использует ту же структуру провайдера. Провайдеры, обеспеченные подписью доверенного лица, могут быть включены в структуру JCE, и новые алгоритмы могут быть добавлены в программный пакет.

Программный интерфейс JCE включает

- Симметричное групповое шифрование, например, DES, RC 2, илиIDEA;
- Симметричное поточное шифрование, например, RC 4;
- Асимметричное шифрование, например, RSA;
- Шифрование с использованием пароля (PBE);
- Согласование ключей;
- Коды аутентификации сообщений (MAC).

Стандартный выпуск программы JAVA 2 SDK, версия 1.5 поставляется с предварительно установленным и зарегистрированным провайдером JCE « SunJCE », который предоставляет следующие криптографические функции:

Использование алгоритмов шифрования

- DES (FIPS PUB 46-1);
- Triple DES и Blowfish в режимах ECB (электронная кодовая книга);
- CBC (сцепление кодовых блоков);
- CFB (программируемый функциональный блок);
- OFB (обратная связь по выходу);
- PCBC (сцепление распространенных кодовых блоков).

> **Примечание**
>
> термины «Triple DES» и «DES-EDE» в данном документе равнозначны.

- Генераторы ключей, используемые для образования ключей, подходящих для алгоритмов стандартов DES, Triple DES, Blowfish, HMAC-MD5 и HMAC-SHA1;
- Использование стандарта MD5 с алгоритмом шифрования с использованием паролей DES-CBC, определенным в стандарте PKCS #5;
- «Фабрики закрытых ключей», обеспечивающие двусторонний переход между зашифрованными объектами ключей DES, Triple DES и PBE и прозрачными представлениями их базового ключевого материала;
- Реализация алгоритма согласования кодов Diffie-Hellman между двумя или более частями;
- Парный генератор кодов Diffie-Hellman для составления общих и частных значений, используемых в алгоритме Diffie-Hellman;
- Параметрический генератор алгоритма Diffie-Hellman;
- «Фабрика ключей» Diffie-Hellman, обеспечивающая двусторонние переходы между зашифрованными ключевыми объектами DiffieHellman и прозрачными представлениями их базового ключевого материала;
- Параметрические диспетчеры алгоритмов для параметров стандартов Diffie-Hellman, DES, Triple DES, Blowfish и PBE;
- Использование алгоритмов хэширования кодов HMAC-MD5 и HMACSHA1, определенных в стандарте RFC 2104;
- Использование схемы дополнения данными, описанной в стандарте PKCS #5;

- Использование собственной реализации ключевого хранилища «JCEKS».

Подробное описание программы JCE представлено на сайте компании Sun

Настройка JSSE

Процедура настройки JSSE проходит в 3 этапа:

- ознакомление с общими требованиями
- интеграция с Internet Explorer
- двусторонняя аутентификация

Общие требования

Полнофункциональная работа с Java апплетами по протоколу GOST TLS с помощью Digt JSSE возможна лишь на платформе win32 с использованием MS Internet Explorer 5 и более поздних версий. В системе должен быть установлен CryptoPro CSP версии 3.6 или 4.0, а также Sun JRE и Java плагин для IE.

Реализация ГОСТ TLS в рамках Java JSSE API с помощью подмены SocketFactory возможна лишь для версий Java 1.5 (и выше) из-за экспортных ограничений, существовавших в предыдущих версиях.

Последовательность действий при интеграции с IE 1) Скопировать библиотеку trusted_java20.jar в каталог $path_to_jre/lib/ext 2) Указать, что при создании SSL соединений должен использоваться DigtSocketFactory класс в файле настроек безопасности \lib\security\java.security

```
#
# Determines the default SSLSocketFactory and SSLServerSocketFactory
# provider implementations for the javax.net.ssl package. If, due to
# export and/or import regulations, the providers are not allowed to be
# replaced, changing these values will produce non-functional
# SocketFactory or ServerSocketFactory implementations.
#
ssl.SocketFactory.provider=com.digt.trusted.jsse.provider.DigtSocketFactory
#ssl.ServerSocketFactory.provider=
```

3) В Control Panel в панели управления java плагином задать параметр java машины:
`Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol`

Двусторонняя аутентификация

Работа с клиентскими сертификатами в Digt JSSE отличается от стандартных способов задания параметров KeyStore в JSSE API в связи с тем, что CryptoPro CSP использует собственный формат хранилища сертификатов и ключевых контейнеров, который не может быть представлен в виде файла.

Для организации защищенных ГОСТ TLS соединений с использованием двусторонней аутентификации Digt JSSE провайдер использует для задания сертификата следующие параметры из системного окружения Java VM, которые могут быть заданы либо с командной строки, используя ключ запуска –D, либо установлены программно с помощью метода System.setProperty():

пример задания из командной строки

```
java –Dcom.digt.trusted.jsse.certFile=file.cer –D com.digt.trusted.jsse.keyPasswd=
```

пример задания из программы на Java

```
System.setProperty("com.digt.trusted.jsse.certFile", "file.cer");
System.setProperty("com.digt.trusted.jsse.keyPasswd", "").
```

Параметр `com.digt.trusted.jsse.certFile` указывает путь к файлу с сертификатом, который должен содержать данные сертификата в формате base64. Параметр `com.digt.trusted.jsse.keyPasswd` содержит необязательный пин-код для ключевого контейнера CryptoPro CSP, который соответствует данному сертификату.

Примеры: Иллюстрацией работы JSSE клиента может служить тестовый класс SSLSocketClient.java из архива примеров, поставляемых с дистрибутивом. Внимание! В данной версии библиотеки существует ограничение, в рамках которого параметры сертификата задаются однократно перед инициализацией провайдера, и их последующее изменение в контексте текущего приложения невозможно без полной выгрузки классов провайдера.

Общая информация

Программный продукт Trusted Java является средством криптографической защиты информации и обеспечивает следующие функциональные возможности:

- создание сертификатов x509;
- формирование запроса на сертификат;
- установка сертификата в хранилище;
- проверка подписи по сертификату;
- работа со списками отозванных сертификатов;
- поддержка CMS;
- поддержка OCSP (проверка статуса сертификата в on-line);
- поддержка TSP (работа со Службой штампов времени).

Trusted Java реализуется в виде JCE провайдера, предоставляющего работу с криптографическими протоколами по алгоритмам ГОСТ. Основу реализованных алгоритмов ГОСТ в Trusted Java составляют криптореализации КриптоПро CSP.

Продукт поддерживает следующие сертифицированные алгоритмы:

- ГОСТ Р 34.11 – основной класс GOST3411Digest;
- ГОСТ Р 34.11-2012 – основные классы GOST3411v12256Digest и GOST3411v12512Digest;
- ГОСТ Р 34.10-94 – класс генерации ключей GOST3410KeyPairGenerator, класс ЭЦП GOST3410Signer (в связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данные классы оставлены только для совместимости с предыдущими версиями);
- ГОСТ Р 34.10-2012 - классы генерации ключей GOST3410v12256KeyPairGenerator и GOST3410v12512KeyPairGenerator, классы ЭЦП GOST3410v12256Signer и GOST3410v12512Signer
- ГОСТ Р 34.10-2001 – класс генерации ключей GOST3410ELKeyPairGenerator, класс ЭЦП GOST3410 EL Signer.
- ГОСТ 28147-89 - основной класс GOST28147Engine;

Наряду с этим, добавлен алгоритм Diffie-Hellman. Разработанные классы наиболее приближены к реализациям классов в JCE.

Сами реализации криптографических алгоритмов в классах GOST отсутствуют. В методах классов вызываются функции из библиотеки с реализациями ГОСТ алгоритмов КриптоПро CSP.

Классы, относящиеся к ГОСТ Р.3410-94, имеют в составе имени GOST3410, классы, относящиеся к ГОСТ Р.3410-2001, – GOST3410EL, классы, относящиеся к ГОСТ Р.3410-2012 – GOST3410v12<256/512>. Помимо этого, некоторые классы (GOST3410EL и GOSTDH) в силу идентичности общих начальных классов ключей и схожей структуры используют одни общие классы с именами GOST3410 (GOSTDH заимствует имена GOST3410, GOSTECDH заимствует имена GOST3410EL).

Строковыми параметрами ключей в GOST являются соответствующие OID.

Отличными от классов той же функциональной группы являются классы GOSTDHAgreement, JDKGOSTKeyStore.

В GOSTDHAgreement при импорте и экспорте сессионный ключ передается шифрованным на общем ключе обмена участников (с дополнительными параметрами для восстановления (дефолтные iv, режим, заполнитель)).

В целях соблюдения совместимости интерфейсов JCE некоторые методы и передаваемые параметры в ГОСТ классах не используются. Подробности даны в примерах и описаниях классов.

Некоторые общие методы

В некоторых классах, где есть необходимость, добавлен метод destroy для уничтожения не использующихся контекстов провайдера, ключей, хешей.

Вызов этого метода происходит непосредственно вручную или же уничтожение не использующихся контекстов происходит при выгрузке djcpбиблиотеки из памяти (в случае, если не использован метод destroy).

Метод public void destroy() вставлен для уничтожения контекстов, создававшихся при работе объекта с библиотекой. Данный метод применяется при необходимости уничтожить контексты до завершения программы.

> **Примечание**
>
> При выгрузке библиотеки из памяти все контексты, создававшиеся GOST объектами, автоматически освобождаются менеджером контекстов, т.е. вызов метода destroy необязателен.

Открыть статью на сайте

Классы, реализующие ГОСТ алгоритмы

GOST28147Engine

GOST3411Digest

GOST3411v12256DIGEST

GOST3411v12512DIGEST

GOST3410

GOST3410ELKeyPairGenerator

GOST3410v12256KeyPairGenerator

GOST3410v12512KeyPairGenerator

GOSTDHKeyPairGenerator

GOSTv12256DHKeyPairGenerator

GOSTv12512DHKeyPairGenerator

GOSTECDHKeyPairGenerator

GOST3410Signer

GOST3410v12256Signer

GOST3410v12512Signer

GOST3410KeyParameters

GOST3410KeyGenParameterSpec

GOST3410PrivateKeyParameters

GOST3410ELPrivateKeyParameters

GOST3410ELPublicKeyParameters

GOST3410v12256PrivateKeyParameters

GOST3410v12256PublicKeyParameters

GOST3410v12512PrivateKeyParameters

GOST3410v12512PublicKeyParameters

GOST3410Parameter

GOST3410v12256Parameter

GOST3410v12512Parameter

GOSTDHAgreement

GOSTSessionKey

GOST3410PrivateKey

GOST3410PublicKey

JDKGOST3410PrivateKey

JDKGOST3410ELPrivateKey

JDKGOST3410ELPublicKey

JDKGOST3410v12256PrivateKey

JDKGOST3410v12256PublicKey

JDKGOST3410v12512PrivateKey

JDKGOST3410v12512PublicKey

JDKGOST3410Signer

JDKGOST3410v12256Signer

JDKGOST3410v12512Signer

GOST3410KeySpec

GOST3410ParameterSpec

GOST3410PublicKeySpec

CAPICertStore

CryptoProCSPKeyStore

TSPHelper

GOST28147Engine

Соответствует	ГОСТ 28147-89
Пакет	com.digt.trusted.crypto.engines
Класс	GOST28147Engine

Совместимость с JCE	Имена методов идентичны другим блочным шифрам jce. Добавлены методы для установки режима шифрования и вектора шифрования. Добавлен метод для инициализации шифра с сессионным ключом. Отличным по логике от шифров jce выступает метод processBytes, которому на вход можно подать сообщение целиком (а не только часть), требующееся для зашифрования или расшифрования. С точки зрения приложения это не оказывает влияние на работу с этим классом, поскольку при работе класс оборачивается в другие классы выбора режимов шифрования, а они в свою очередь умеют работать с GOST28147Engine также как и с другими классами блочных шифров (см. примеры использования класса). Padding установлен в RANDOM_PADDING, поэтому нет смысла выбирать другой заполнитель, в java везде используется тип “/NoPadding”
Описание класса	Шифрование по алгоритму ГОСТ 28147-89 Реализует интерфейс BlockCipher
Методы класса	public void init(boolean encrypting, CipherParameters params) – инициализирует шифр ГОСТ 28147-89. Параметры: encrypting – если «true» - шифр инициализируется для зашифрования, если «false» - для расшифрования. params – параметры, необходимые для установки шифра. В качестве параметров передается ключевая информация (набор заданных байт или сессионный ключ). public void init(boolean forEncryption, GOSTSessionKey seskey) – инициализирует шифр ГОСТ 28147-89. Параметры: encrypting – если «true» - шифр инициализируется для зашифрования, если «false » - для расшифрования. seskey – сессионный ключ формата GOSTSessionKey. public int processBlock (byte[] in, int inOff, byte[] out, int outOff) – выполняет шифрование одного блока данных из входного массива и выводит результат в выходной массив. Параметры: in – массив, содержащий входной блок данных. inOff – смещение во входном массиве, с которого начинаются байты. out – выходной массив, в который будут скопированы данные. outOff – смещение в выходном массиве, с которого начнутся результирующие байты. Возвращает размер одного блока в ГОСТ 28147-89. public int processBlocks (byte[] in, int inOff, byte[] out, int outOff, int blocks) – выполняет шифрование

	указанного количества входных блоков данных и выводит результат в выходной массив. Метод добавлен для повышения производительности – позволяет обрабатывать несколько блоков данных за один вызов. Параметры: blocks – количество обрабатываемых блоков данных. остальные параметры см. public int processBlock(byte[] in, int inOff, byte[] out, int outOff). Возвращает размер блока в ГОСТ 28147-89, умноженный на количество блоков, переданных для обработки. public void processBytes (byte[] in, int inOff, int length, byte[] out, int outOff) – выполняет шифрование указанного количества байтов из входного массива и выводит результат в выходной массив. Метод добавлен для возможности обработки данных блоками, размер которых не кратен размеру блока в ГОСТ 28147-89, из классов режимов, использовать его напрямую не рекомендуется! Параметры: length – количество байт для обработки (для режимов “CFB” и “OFB” –любое положительное число, а для режима “ECB” должно быть кратно размеру блока в ГОСТ 29147-89). остальные параметры см. public int processBlock(byte[] in, int inOff, byte[] out, int outOff). public String getAlgorithmName() – возвращает имя алгоритма блочного шифра. public void setMode (String mode) – устанавливает режим шифрования (используется для установки блочного режима “ECB”). public void setMode (String mode, byte[] IV) – устанавливает режим шифрования (используется для установки потоковых режимов “CFB” и “OFB”) и начального вектора инициализации. public String getMode() – возвращает используемый режим шифрования. public int getBlockSize() – возвращает размер одного блока при шифровании в ГОСТ 28147-89 (количество байт)
Примеры	см. com.digt.trusted.jce.provider.test.GOST28147Test

GOST3411Digest

Соответствует	интерфейс хеширования по ГОСТ Р 34.11-94
Пакет	com.digt.trusted.crypto
Класс	GOST3411Digest

Описание класса	Хэширование данных в соответствии с алгоритмами ГОСТ Р 34.11-94
Примеры	см. com.digt.trusted.jce.provider.test.GOST3411Test

GOST3411v12256Digest

Соответствует	интерфейс хеширования по ГОСТ Р 34.11-2012 (256)
Пакет	com.digt.trusted.crypto
Класс	GOST3411v12256Digest
Описание класса	Хэширование данных в соответствии с алгоритмами ГОСТ Р 34.11-2012
Примеры	см. com.digt.trusted.jce.provider.test.GOST3411v12Test

GOST3411v12512Digest

Соответствует	интерфейс хеширования по ГОСТ Р 34.11-2012 (512)
Пакет	com.digt.trusted.crypto
Класс	GOST3411v12512Digest
Описание класса	Хэширование данных в соответствии с алгоритмами ГОСТ Р 34.11-2012
Примеры	см. com.digt.trusted.jce.provider.test.GOST3411v12Test

GOST3410

Соответствует	интерфейс подписи ГОСТ 3410
Пакет	com.digt.trusted.crypto
Класс	GOST3410
Совместимость с JCE	В методах generateSignature и verifySignature параметр message передается как полное сообщение, а не хеш сообщение. Также подпись не состоит из двух чисел r, s, она представляет собой единый блок данных длиной 64 байта (512 бит), первая половина r, вторая s.

Описание класса	Интерфейс описывает общую структуру классов подписи ГОСТ 3410
Методы класса	public void init (boolean forSigning, CipherParameters param) – инициализирует класс подписчика для генерации подписи или проверки подписи. Параметры: forSigning – если «true» - генерация подписи, в ином случае - проверка.param – ключевые параметры для подписи (это закрытый или открытый ключ). public byte[] generateSignature(byte[] message) – подписывает переданное сообщение. Параметры: message – сообщение для подписи. Возвращает блок данных, являющийся подписанным сообщением (или просто ЭЦП). public boolean verifySignature (byte[] message, byte[] sign) – проверяет истинность подписанного сообщения. Параметры: message – сообщение, которое было подписано. sign – значение подписи.
Примеры	см. com.digt.trusted.crypto.signers.GOST3410Signer com.digt.trusted.crypto.signers.GOST3410ELSigner com.digt.trusted.crypto.signers.GOST3410v12256Signer com.digt.trusted.crypto.signers.GOST3410v12512Signer com.digt.trusted.jce.provider.JDKGOST3410Signer com.digt.trusted.jce.provider.JDKGOST3410ELSigner Com.digt.trusted.jce.provider.JDKGOST3410v12256Signer com.digt.trusted.jce.provider.JDKGOST3410v12512Signer

GOST3410ELKeyPairGenerator

Соответствует	Генератор ключей по ГОСТ 3410-2001
Пакет	com.digt.trusted.crypto.generators
Класс	GOST3410ELKeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOST3410ELKeyPairGenerator не имеет алгебраических параметров. Вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа

	не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2001, сохраняемая в ключевом контейнере. Предназначена для ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator. Класс GOST3410ELKeyPairGenerator сходен с GOST 3410 KeyPairGenerator за исключением возвращаемого открытого ключа (в 512 бит)
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару по ГОСТ 3410-2001, открытый ключ длиной 512 бит. Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410ELTest

GOST3410v12256KeyPairGenerator

Соответствует	Генератор ключей по ГОСТ 3410-2012 (256)
Пакет	com.digt.trusted.crypto.generators
Класс	GOST3410v12256KeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOST3410v12256KeyPairGenerator не имеет алгебраических параметров. Вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)

Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2012, сохраняемая в ключевом контейнере. Предназначена для ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator. Класс GOST3410v12256KeyPairGenerator сходен с GOST3410KeyPairGenerator за исключением возвращаемого открытого ключа
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару по ГОСТ 3410-2012 (256). Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test

GOST3410v12512KeyPairGenerator

Соответствует	Генератор ключей по ГОСТ 3410-2012 (512)
Пакет	com.digt.trusted.crypto.generators
Класс	GOST3410v12512KeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOST3410v12512KeyPairGenerator не имеет алгебраических параметров. Вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2012, сохраняемая в ключевом контейнере. Предназначена для ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator. Класс GOST3410v12512KeyPairGenerator сходен с

	GOST 3410 KeyPairGenerator за исключением возвращаемого открытого ключа
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару по ГОСТ 3410-2012, с длиной 512. Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test

GOSTDHKeyPairGenerator

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-94
Пакет	com.digt.trusted.crypto.generators
Класс	GOSTDHKeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOSTDHKeyPairGenerator не имеет алгебраических параметров. Вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-94, сохраняемая в ключевом контейнере. Она предназначена для обмена сессионными ключами и ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator

Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, в котором передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-94, открытый ключ длиной 1024 бит. Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи

GOSTDHKeyPairGenerator

Соответствует	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-2012 256
Пакет	com.digt.trusted.crypto.generators
Класс	GOSTv12256DHKeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOSTv12256DHKeyPairGenerator не имеет алгебраических параметров, вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается, как есть, с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2012, сохраняемая в ключевом контейнере. Она предназначена для обмена сессионными ключами и ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-2012 с

	длиной 256. Возвращает AsymmetricCipherKeyPai r – содержит сгенерированные ключи

GOSTv12512DHKeyPairGenerator

Соответствует	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-2012 512
Пакет	com.digt.trusted.crypto.generators
Класс	GOSTv12512DHKeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOSTv12512DHKeyPairGenerator не имеет алгебраических параметров, вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается, как есть, с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2012, сохраняемая в ключевом контейнере. Она предназначена для обмена сессионными ключами и ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-2012 с длиной 512. Возвращает AsymmetricCipherKeyPai r – содержит сгенерированные ключи

GOSTECDHKeyPairGenerator

Соответствует	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-2001
Пакет	com.digt.trusted.crypto.generators
Класс	GOSTECDHKeyPairGenerator
Совместимость с JCE	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то GOSTECDHKeyPairGenerator не имеет алгебраических параметров, вместо параметра в класс передается имя ключевого контейнера, в котором генерируется ключевая пара. При возврате значения ключевой пары, значение закрытого ключа не возвращается, а передается только имя ключевого контейнера, в котором этот ключ хранится. Открытый ключ возвращается, как есть, с заданными параметрами (строковыми константами, а не числовыми, как в других алгоритмах)
Описание класса	Производится ключевая пара согласно ГОСТ Р 34.10-2001, сохраняемая в ключевом контейнере. Она предназначена для обмена сессионными ключами и ЭЦП. Реализует интерфейс AsymmetricCipherKeyPairGenerator
Методы класса	public void init (KeyGenerationParameters param) – задает начальные параметры генератора. Параметры: param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара. public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-2001, открытый ключ длиной 512 бит. Возвращает AsymmetricCipherKeyPai r – содержит сгенерированные ключи

GOST3410Signer

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Электронная цифровая подпись ГОСТ 3410-94
Пакет	com.digt.trusted.crypto.signers

Класс	GOST3410Signer
Совместимость с JCE	совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-94. Реализует интерфейс GOST3410.
Методы класса	public void init (boolean forSigning, CipherParameters param) – инициализирует класс подписчика для генерации подписи или проверки подписи. Параметры: forSigning – если «true» - генерация подписи, в ином случае - проверка. param – ключевые параметры для подписи (закрытый или открытый ключ). public byte[] generateSignature(byte[] message) – подписывает переданное сообщение. Параметры: message – хеш сообщения для подписи. Возвращает подписанное сообщение. public boolean verifySignature (byte[] message, byte[] sign) – проверяет истинность подписанного сообщения. Параметры: message – хеш сообщения, которое было подписано. sign – значение подписи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410Test com.digt.trusted.jce.provider.test.CertTest – проверка подписи сертификата

GOST3410v12256Signer

Соответствует	Электронная цифровая подпись ГОСТ 3410-2012 (256)
Пакет	com.digt.trusted.crypto.signers
Класс	GOST3410v12256Signer
Совместимость с JCE	совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-2012. Реализует интерфейс GOST3410.
Методы класса	public void init (boolean forSigning, CipherParameters param) – инициализирует класс подписчика для генерации подписи или проверки подписи.

	Параметры: forSigning – если «true» - генерация подписи, в ином случае - проверка. param – ключевые параметры для подписи (закрытый или открытый ключ). public byte[] generateSignature(byte[] message) – подписывает переданное сообщение. Параметры: message – хеш сообщения для подписи. Возвращает подписанное сообщение. public boolean verifySignature (byte[] message, byte[] sign) – проверяет истинность подписанного сообщения. Параметры: message – хеш сообщения, которое было подписано. sign – значение подписи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test com.digt.trusted.jce.provider.test.CertTest – проверка подписи сертификата

GOST3410v12512Signer

Соответствует	Электронная цифровая подпись ГОСТ 3410-2012 (512)
Пакет	com.digt.trusted.crypto.signers
Класс	GOST3410v12512Signer
Совместимость с JCE	совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-2012. Реализует интерфейс GOST3410.
Методы класса	public void init (boolean forSigning, CipherParameters param) – инициализирует класс подписчика для генерации подписи или проверки подписи. Параметры: forSigning – если «true» - генерация подписи, в ином случае - проверка. param – ключевые параметры для подписи (закрытый или открытый ключ). public byte[] generateSignature(byte[] message) – подписывает переданное сообщение. Параметры: message – хеш сообщения для подписи. Возвращает подписанное сообщение.

	public boolean verifySignature (byte[] message, byte[] sign) – проверяет истинность подписанного сообщения. Параметры: message – хеш сообщения, которое было подписано. sign – значение подписи
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test com.digt.trusted.jce.provider.test.CertTest – проверка подписи сертификата

GOST3410KeyParameters

Соответствует	Ключевые параметры для алгоритмов ГОСТ 3410
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410KeyParameters
Совместимость с JCE	Аналогичен другим подобным классам
Описание класса	Служит контейнером для ключевых параметров из КриптоПро CSP. Общий класс в классах GOST3410, GOST3410EL, GOSTDH. Расширяет класс AsymmetricKeyParameter
Методы класса	protected GOST3410KeyParameters(boolean isPrivate, GOST3410Parameters params) – стандартный конструктор, создает класс с параметрами. Параметры: isPrivate – устанавливает флаг типа параметров ключа. params – ключевые параметры. public GOST3410Parameters getParameters() – возвращает параметры класса. public boolean equals(Object obj) – сравнивает 2 объекта GOST3410KeyParameters
Примеры	см. com.digt.trusted.crypto.params.GOST3410PublicKeyParameters com.digt.trusted.crypto.signers.GOST3410Signer com.digt.trusted.crypto.signers.GOST3410ELSigner и др.

GOST3410KeyGenParameterSpec

Соответствует	Параметр при генерации ключевых пар ГОСТ 3410
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410KeyGenParameterSpec
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит имя ключевого контейнера, в котором хранится ключевая пара. Реализует интерфейс AlgorithmParameterSpec
Методы класса	public GOST3410KeyGenParameterSpec(String container) – стандартный конструктор Параметры: container – имя контейнера, в котором находится ключевая пара. public String getContainer() – возвращает имя контейнера
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410Test com.digt.trusted.jce.provider.test.GOST3410ELTest

GOST3410PrivateKeyParameters

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Закрытый ключ с параметрами ГОСТ 3410-94
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410PrivateKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	public GOST3410PrivateKeyParameters(String x, GOST3410Parameters params) – стандартный конструктор. Параметры:

	x – имя контейнера, в котором находится ключевая пара. params – параметры к ключу. public String getX() – возвращает закрытый ключ (имя контейнера, в котором он лежит). public boolean equals(Object obj) – сравнивает 2 объекта GOST3410PrivateKeyParameters
Примеры	см. com.digt.trusted.crypto.generators.GOST3410KeyPairGenerator com.digt.trusted.crypto.signers. GOST3410Signer com.digt.trusted.jce.provider.JDKGOST3410PrivateKey и др.

GOST3410ELPrivateKeyParameters

Соответствует	Закрытый ключ с параметрами ГОСТ 3410-2001
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410ELPrivateKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	public GOST3410ELPrivateKeyParameters(String x, GOST3410Parameters params) – стандартный конструктор. Параметры: x – имя контейнера, в котором находится ключевая пара. params – параметры к ключу. public String getX () – возвращает закрытый ключ (имя контейнера, в котором он хранится). public boolean equals(Object obj) – сравнивает 2 объекта GOST3410ELPrivateKeyParameters
Примеры	см. com.digt.trusted.crypto.generators.GOST3410ELKeyPairGenerator com.digt.trusted.crypto.signers.GOST3410ELSigner com.digt.trusted.jce.provider.JDKGOST3410ELPrivateKey и др.

GOST3410ELPublicKeyParameters

Соответствует	Открытый ключ с параметрами ГОСТ 3410-2001
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410ELPublicKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит открытый ключ с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	public GOST3410ELPublicKeyParameters (byte[] y, GOST 3410 Parameters params) – конструктор создает открытый ключ с параметрами. Параметры: y – открытый ключ в виде блока байт. params – параметры открытого ключа. public byte[] getY() – возвращает открытый ключ (без параметров)
Примеры	см. com.digt.trusted.crypto.generators.GOST3410ELKeyPairGenerator com.digt.trusted.crypto.signers.GOST3410v12256Signer com.digt.trusted.jce.provider.JDKGOST3410v12256PublicKey и др.

GOST3410v12256PrivateKeyParameters

Соответствует	Закрытый ключ с параметрами ГОСТ 3410-2012 (256)
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410v12256PrivateKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	

	public GOST3410v12256PrivateKeyParameters(String x, GOST3410Parameters params) – стандартный конструктор. Параметры: x – имя контейнера, в котором находится ключевая пара. params – параметры к ключу. public String getX () – возвращает закрытый ключ (имя контейнера, в котором он хранится). public boolean equals(Object obj) – сравнивает 2 объекта GOST3410v12256PrivateKeyParameters
Примеры	см. com.digt.trusted.crypto.generators.GOST3410v12256KeyPairGenerator com.digt.trusted.crypto.signers.GOST3410v12256Signer com.digt.trusted.jce.provider.JDKGOST3410v12256PrivateKey и др.

GOST3410v12256PublicKeyParameters

Соответствует	Открытый ключ с параметрами ГОСТ 3410-2012 (256)
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410v12256PublicKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит открытый ключ с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	public GOST3410v12256PublicKeyParameters (byte[] y, GOST 3410 Parameters params) – конструктор создает открытый ключ с параметрами. Параметры: y – открытый ключ в виде блока байт. params – параметры открытого ключа. public byte[] getY() – возвращает открытый ключ (без параметров)
Примеры	см. com.digt.trusted.crypto.generators.GOST3410v12256KeyPairGenerator com.digt.trusted.crypto.signers.GOST3410v12256Signe

	r com.digt.trusted.jce.provider.JDKGOST3410v12256PublicKey и др.

GOST3410v12512PrivateKeyParameters

Соответствует	Закрытый ключ с параметрами ГОСТ 3410-2012 (512)
Пакет	com.digt.trusted.crypto.params
Класс	GOST3410v12512PrivateKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами. Расширяет класс GOST3410KeyParameters
Методы класса	public GOST3410v12512PrivateKeyParameters(String x, GOST3410Parameters params) – стандартный конструктор. Параметры: x – имя контейнера, в котором находится ключевая пара. params – параметры к ключу. public String getX () – возвращает закрытый ключ (имя контейнера, в котором он хранится). public boolean equals(Object obj) – сравнивает 2 объекта GOST3410v12512PrivateKeyParameters
Примеры	см. com.digt.trusted.crypto.generators.GOST3410v12512KeyPairGenerator com.digt.trusted.crypto.signers.GOST3410v12512Signer com.digt.trusted.jce.provider.JDKGOST3410v12512PrivateKey и др.

GOST3410v12512PublicKeyParameters

Соответствует	Открытый ключ с параметрами ГОСТ 3410-2012 (512)
Пакет	com.digt.trusted.crypto.params

Класс	GOST3410v12512PublicKeyParameters
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит открытый ключ с дополнительными параметрами. <brРасширяет класс GOST3410KeyParameters
Методы класса	public GOST3410v12512PublicKeyParameters (byte[] y, GOST 3410 Parameters params) – конструктор создает открытый ключ с параметрами. Параметры: y – открытый ключ в виде блока байт. params – параметры открытого ключа. public byte[] getY() – возвращает открытый ключ (без параметров)
Примеры	см. com.digt.trusted.crypto.generators.GOST3410v12512KeyPairGenerator com.digt.trusted.crypto.signers.GOST3410v12512Signer com.digt.trusted.jce.provider.JDKGOST3410v12512PublicKey и др.

GOST3410Parameter

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Параметры открытого ключа в asn1 по ГОСТ3410-94
Пакет	com.digt.trusted.asn1.x509
Класс	GGOST3410Parameter
Совместимость с JCE	Без существенных отличий
Описание класса	Служит для хранения параметров открытого ключа вида asn 1, передается в SubjectPublicKeyInfo в классе GOST3410Parameter Реализует интерфейс DEREncodable
Методы класса	public static GOST3410Parameter getInstance(ASN1TaggedObject obj, boolean explicit) – no comments. public static GOST3410Parameter getInstance(Object obj) – no comments. public GOST3410Parameter(String p1, String p2) – конструктор с двумя параметрами на входе.

	public GOST3410Parameter(ASN1Sequence seq) – no comments. public String getP1() – возвращает первый параметр. public String getP2() – возвращает второй параметр. public DERObject getDERObject() – возвращает объект параметров в DERObject
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410PublicKey

GOST3410v12256Parameter

Соответствует	Параметры открытого ключа в asn1 по ГОСТ3410-2012
Пакет	com.digt.trusted.asn1.x509
Класс	GOST3410v12256Parameter
Совместимость с JCE	Без существенных отличий
Описание класса	Служит для хранения параметров открытого ключа вида asn 1, передается в SubjectPublicKeyInfo в классе GOST3410Parameter Реализует интерфейс DEREncodable
Методы класса	public static GOST3410v12256Parameter getInstance(ASN1TaggedObject obj, boolean explicit) – no comments. public static GOST3410v12256Parameter getInstance(Object obj) – no comments. public GOST3410v12256Parameter(String p1, String p2) – конструктор с двумя параметрами на входе. public GOST3410v12256Parameter(ASN1Sequence seq) – no comments. public String getP1() – возвращает первый параметр. public String getP2() – возвращает второй параметр. public DERObject getDERObject() – возвращает объект параметров в DERObject
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410PublicKey

GOST3410v12512Parameter

Соответствует	

	Параметры открытого ключа в asn1 по ГОСТ3410-2012
Пакет	com.digt.trusted.asn1.x509
Класс	GOST3410v12512Parameter
Совместимость с JCE	Без существенных отличий
Описание класса	Служит для хранения параметров открытого ключа вида asn1, передается в SubjectPublicKeyInfo в классе GOST3410Parameter Реализует интерфейс DEREncodable
Методы класса	public static GOST3410v12512Parameter getInstance(ASN1TaggedObject obj, boolean explicit) – no comments. public static GOST3410v12512Parameter getInstance(Object obj) – no comments. public GOST3410v12512Parameter(String p1, String p2) – конструктор с двумя параметрами на входе. public GOST3410v12512Parameter(ASN1Sequence seq) – no comments. public String getP1() – возвращает первый параметр. public String getP2() – возвращает второй параметр.
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410v12512PublicKey

GOSTDHAgreement

Соответствует	Основа для алгоритма Diffie-Hellman
Пакет	com.digt.trusted.crypto.agreement
Класс	GOSTDHAgreement
Совместимость с JCE	Присутствуют те же методы, что и в классах той же группы (Agreement). Добавлены новые методы с новыми функционалами. Класс не вычисляет общий закрытый ключ, в нем происходит экспорт (импорт) сессионного ключа шифрованного на общем ключе обмена
Описание класса	В классе сохраняется собственный закрытый ключ (init) Помещается открытый ключ отправителя (calculateAgreement) Экспортируется сессионный ключ

	(exportSessionKey) Импортируется сессионный ключ (importSessionKey) Подробности ниже в описании методов
Методы класса	public void init (CipherParameters param) – сохраняется в классе собственный закрытый ключ, переданный в классе параметров (имя ключевого контейнера). public void init (Key key) – сохраняется закрытый ключ, переданный в классе privatekey. public BigInteger calculateAgreement(GOST3410PublicKeyParameters pub) – сохраняет. Возвращает всегда null. (BigInteger – в целях совместимости). public CipherParameters getPrivateKeyParameters() – возвращает параметры закрытого ключа. Функция неактуальна, оставлена в целях совместимости. public GOST3410PublicKeyParameters getPublicKeyParameters() – возвращает открытый ключ с параметрами. public String getContainer() – возвращает имя контейнера. public byte[] exportSessionKey(GOSTSessionKey gsk) – экспортирует сессионный ключ. Параметры: gsk – структура сессионного ключа на экспорт Возвращает блок данных, готовых к передаче на сторону получателя. public GOSTSessionKey importSessionKey(byte[] impkey) – импортирует сессионный ключ. Параметры: impkey – принятый блок сессионного ключа. Возвращает структуру восстановленного сессионного ключа

GOSTSessionKey

Соответствует	Сессионный ключ ГОСТ 28147
Пакет	com.digt.trusted.crypto.GOSTSessionKey
Класс	GOSTSessionKey
Совместимость с JCE	Без существенных отличий
Описание класса	Класс создан для хранения, генерации и передачи сессионного ключа (с параметрами). Реализует интерфейс SecretKey

Методы класса	public GOSTSessionKey() – пустой конструктор. public GOSTSessionKey (byte[] EncodedObject) – конструктор, создающий из пришедшего блока EncodedObject полноценный класс (импортирует блок в структуру полей класса). public byte[] getEncoded() – кодирует поля класса с возможностью дальнейшего экспорта класса. public void genKey (String containerName) – генерирует сессионный ключ, шифруемый ключом обмена. public void setkeyBlob (byte[] keyBlob) – устанавливает блок, содержащий шифрованный сессионный ключ. public void setKP_IV (byte[] KP_IV) – устанавливает начальный вектор инициализации. public void setKP_PADDING (byte[] KP_PADDING) – устанавливает умалчиваемый режим заполнения. public void setKP_MODE (byte[] KP_MODE) – устанавливает умалчиваемый режим шифрования. public void sethProv (int hProv) – устанавливает хендл провайдера. public void sethKey(int hKey) – устанавливает хендл ключа. public byte[] getkeyBlob() – возвращает блок, содержащий шифрованный сессионный ключ. public byte[] getKP_IV() – возвращает начальный вектор инициализации. public byte[] getKP_PADDING() – возвращает умалчиваемый режим заполнения. public byte[] getKP_MODE() – возвращает умалчиваемый режим шифрования. public int gethProv() – возвращает хендл провайдера. public int gethKey() – возвращает хендл ключа. public String getAlgorithm() – no comments. public String getFormat() – no comments. public void destroy() – см. «Некоторые общие методы»
Примеры	см. com.digt.trusted.crypto.agreement.GOSTDHAgreement

GOST3410PrivateKey

Соответствует	Общий интерфейс для закрытых ключей классов GOST3410
Пакет	com.digt.trusted.jce.interfaces
Класс	GOST3410PrivateKey

Совместимость с JCE	no comments
Описание класса	Расширяет интерфейс PrivateKey
Методы класса	public String getX () – возвращает ссылку на закрытый ключ

GOST3410PublicKey

Соответствует	Общий интерфейс для открытых ключей классов GOST3410
Пакет	com.digt.trusted.jce.interfaces
Класс	GOST3410PublicKey
Совместимость с JCE	no comments
Описание класса	Расширяет интерфейсы GOST3410Key, PublicKey
Методы класса	public byte[] getY() – возвращает открытый ключ

JDKGOST3410PrivateKey

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Закрытый ключ по ГОСТ 3410-94 на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410PrivateKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PrivateKeyy
Методы класса	public JDKGOST3410PrivateKey(GOST3410PrivateKey key) – инкапсулирует закрытый ключ типа GOST3410PrivateKey в объект. public JDKGOST3410PrivateKey(GOST3410PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeySpec в объект.

	public JDKGOST3410PrivateKey (GOST3410PrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeyParameters в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public String getX() – возвращает закрытый ключ. public byte[] getEncoded() – возвращает всегда null
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410Signer com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH

JDKGOST3410ELPrivateKey

Соответствует	Закрытый ключ по ГОСТ 3410-2001 на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410ELPrivateKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PrivateKey
Методы класса	public JDKGOST3410ELPrivateKey(GOST3410PrivateKey key) – инкапсулирует закрытый ключ типа GOST3410 PrivateKey в объект. public JDKGOST3410ELPrivateKey(GOST3410PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeySpec в объект. public JDKGOST3410ELPrivateKey(GOST3410ELPrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410ELPrivateKeyParameters в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public String getX () – возвращает закрытый ключ. public byte[] getEncoded() – возвращает всегда null

Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410ELSigner com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410EL com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH

JDKGOST3410ELPublicKey

Соответствует	Открытый ключ по ГОСТ 3410-2001 на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410ELPublicKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе открытый ключ с параметрами, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PublicKey
Методы класса	public JDKGOST3410ELPublicKey(GOST3410PublicKey key) – инкапсулирует открытый ключ типа GOST3410PublicKey в объект. public JDKGOST3410ELPublicKey(GOST3410PublicKeySpec spec) – инкапсулирует открытый ключ типа GOST3410PublicKeySpec в объект. public JDKGOST3410ELPublicKey(GOST3410ELPublicKeyParameters params) – инкапсулирует открытый ключ типа GOST 3410 ELPublicKeyParameters в объект. public JDKGOST3410ELPublicKey(SubjectPublicKeyInfo info) – инкапсулирует открытый ключ типа SubjectPublicKeyInfo в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public byte[] getY() – возвращает открытый ключ. public byte[] getEncoded() – декодирует объект в блок байт. public GOST3410ParameterSpec getParams() – возвращает параметры ключа. public String toString() – no comments
Примеры	

	см. com.digt.trusted.jce.provider.JDKGOST3410ELSigner com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410EL com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH

JDKGOST3410v12256PrivateKey

Соответствует	Закрытый ключ по ГОСТ 3410-2012 (256) на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12256PrivateKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PrivateKey
Методы класса	public JDKGOST3410v12256PrivateKey(GOST3410PrivateKey key) – инкапсулирует закрытый ключ типа GOST3410PrivateKey в объект. public JDKGOST3410v12256PrivateKey(GOST3410PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeySpec в объект. public JDKGOST3410v12256PrivateKey (GOST3410v12256PrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410v12256PrivateKeyParameters в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public String getX () – возвращает закрытый ключ. public byte[] getEncoded()
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410v12256Signer com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410v12256 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTv12256DH

JDKGOST3410v12256PublicKey

Соответствует	Открытый ключ по ГОСТ 3410-2012 (256) на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12256PublicKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе открытый ключ с параметрами, используется во взаимодействии с криптопровайдером Реализует интерфейс GOST3410PublicKey
Методы класса	public JDKGOST3410v12256PublicKey(GOST3410PublicKey key) – инкапсулирует открытый ключ типа GOST3410PublicKey в объект. public JDKGOST3410v12256PublicKey(GOST3410PublicKeySpec spec) – инкапсулирует открытый ключ типа GOST3410PublicKeySpec в объект. public JDKGOST3410v12256PublicKey(GOST3410v12256PublicKeyParameters params) – инкапсулирует открытый ключ типа GOST3410v12256PublicKeyParameters в объект. public JDKGOST3410v12256PublicKey(SubjectPublicKeyInfo info) – инкапсулирует открытый ключ типа SubjectPublicKeyInfo в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public byte[] getY() – возвращает открытый ключ. public byte[] getEncoded() – декодирует объект в блок байт. public GOST3410ParameterSpec getParams() – возвращает параметры ключа. public String toString() – no comments
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410v12256Signer com.digt.trusted.jce.provider. JDKKeyPairGenerator.GOST3410v12256 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTv12256DH

JDKGOST3410v12512PrivateKey

Соответствует	Закрытый ключ по ГОСТ 3410-2012 (512) на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12256PublicKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PrivateKey
Методы класса	public JDKGOST3410v12512PrivateKey(GOST3410PrivateKey key) – инкапсулирует закрытый ключ типа GOST3410PrivateKey в объект. public JDKGOST3410v12512PrivateKey(GOST3410PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeySpec в объект. public JDKGOST3410v12512PrivateKey(GOST3410v12512PrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410v12512PrivateKeyParameters в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public String getX () – возвращает закрытый ключ. public byte[] getEncoded()
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410v12512Signer com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410v12512 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTv12512DH

JDKGOST3410v12512PublicKey

Соответствует	Открытый ключ по ГОСТ 3410-2012 (512) на уровне криптопровайдера

Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12512PublicKey
Совместимость с JCE	Без существенных отличий
Описание класса	Хранит в классе открытый ключ с параметрами, используется во взаимодействии с криптопровайдером. Реализует интерфейс GOST3410PublicKey
Методы класса	public JDKGOST3410v12512PublicKey(GOST3410PublicKey key) – инкапсулирует открытый ключ типа GOST3410PublicKey в объект. public JDKGOST3410v12512PublicKey(GOST3410PublicKeySpec spec) – инкапсулирует открытый ключ типа GOST3410PublicKeySpec в объект. public JDKGOST3410v12512PublicKey(GOST3410v12512PublicKeyParameters params) – инкапсулирует открытый ключ типа GOST3410v12512PublicKeyParameters в объект. public JDKGOST3410v12512PublicKey(SubjectPublicKeyInfo info) – инкапсулирует открытый ключ типа SubjectPublicKeyInfo в объект. public String getAlgorithm() – возвращает алгоритм. public String getFormat() – возвращает тип хранения и передачи ключа. public byte[] getY() – возвращает открытый ключ. public byte[] getEncoded() – декодирует объект в блок байт. public GOST3410ParameterSpec getParams() – возвращает параметры ключа. public String toString() – no comments
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410v12512Signer com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410v12512 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTv12512DH

JDKGOST3410Signer

В связи с тем, что стандарт ГОСТ Р 34.10-94 устарел, данный класс оставлен только для совместимости с предыдущими версиями.

Соответствует	Электронная цифровая подпись ГОСТ 3410-94 на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410Signer
Совместимость с JCE	Совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-94. Расширяет класс Signature
Методы класса	protected JDKGOST3410Signer(String name, Digest digest, GOST3410signer) – инкапсуляция определяющих параметров при цифровой подписи. Параметры: name – константное имя « GOST3411withGOST3410» digest – класс, использующийся для получения хеша сообщения перед подписью. signer – класс подписчика сообщений(GOST3410Signer). protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки ЭЦП. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки подписи. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. Параметры: privateKey – передается закрытый ключ. random – не используется (передается null). protected void engineInitSign (PrivateKey privateKey) – инициализирует класс для проверки ЭЦП. В параметре передается закрытый ключ. protected void engineUpdate (byte b) – «сохраняет в класс» байт подписанного сообщения. protected void engineUpdate (byte[] b, int off, int len) – «сохраняет в класс» блок байт подписанного сообщения. Начиная с off длиной len. protected byte[] engineSign() – подписывает сообщение и возвращает его ЭЦП. protected boolean engineVerify(byte[] sigBytes) – проверяет истинность подписанного сообщения, где sigBytes – блок подписи. Возвращает true – если ЭЦП верна, в ином случае - false. protected void engineSetParameter(AlgorithmParameterSpec params)

	– функция не применяется. Выдает UnsupportedOperationException. protected void engineSetParameter(String param, Object value) – функция не применяется. Выдает UnsupportedOperationException. protected Object engineGetParameter(String param) – функция не применяется. Выдает UnsupportedOperationException
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410Test

JDKGOST3410v12256Signer

Соответствует	Электронная цифровая подпись ГОСТ 3410-2012 (256) на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12256Signer
Совместимость с JCE	Совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-2012. Расширяет класс Signature
Методы класса	protected JDKGOST3410v12256Signer(String name, Digest digest, GOST3410signer) – инкапсуляция определяющих параметров при цифровой подписи. Параметры: name – константное имя « GOST3411withGOST3410» digest – класс, использующийся для получения хеша сообщения перед подписью. signer – класс подписчика сообщений(GOST3410Signer). protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки ЭЦП. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки подписи. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. Параметры: privateKey – передается закрытый ключ. random – не используется (передается null). protected void engineInitSign (PrivateKey privateKey) –

	инициализирует класс для проверки ЭЦП. В параметре передается закрытый ключ. protected void engineUpdate (byte b) – «сохраняет в класс» байт подписанного сообщения. protected void engineUpdate (byte[] b, int off, int len) – «сохраняет в класс» блок байт подписанного сообщения. Начиная с off длиной len. protected byte[] engineSign() – подписывает сообщение и возвращает его ЭЦП. protected boolean engineVerify(byte[] sigBytes) – проверяет истинность подписанного сообщения, где sigBytes – блок подписи. Возвращает true – если ЭЦП верна, в ином случае - false. protected void engineSetParameter(AlgorithmParameterSpec params) – функция не применяется. Выдает UnsupportedOperationException. protected void engineSetParameter(String param, Object value) – функция не применяется. Выдает UnsupportedOperationException. protected Object engineGetParameter(String param) – функция не применяется. Выдает UnsupportedOperationException
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test

JDKGOST3410v12512Signer

Соответствует	Электронная цифровая подпись ГОСТ 3410-2012 (512) на уровне криптопровайдера
Пакет	com.digt.trusted.jce.provider
Класс	JDKGOST3410v12512Signer
Совместимость с JCE	Совместим
Описание класса	Подпись сообщения и проверка подписи по ГОСТ 3410-2012. Расширяет класс Signature
Методы класса	protected JDKGOST3410v12512Signer(String name, Digest digest, GOST3410signer) – инкапсуляция определяющих параметров при цифровой подписи. Параметры: name – константное имя « GOST3411withGOST3410» digest – класс, использующийся для получения хеша сообщения перед подписью. signer – класс подписчика

	сообщений(GOST3410Signer). protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки ЭЦП. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки подписи. В параметре передается открытый ключ. protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи. Параметры: privateKey – передается закрытый ключ. random – не используется (передается null). protected void engineInitSign (PrivateKey privateKey) – инициализирует класс для проверки ЭЦП. В параметре передается закрытый ключ. protected void engineUpdate (byte b) – «сохраняет в класс» байт подписанного сообщения. protected void engineUpdate (byte[] b, int off, int len) – «сохраняет в класс» блок байт подписанного сообщения. Начиная с off длиной len. protected byte[] engineSign() – подписывает сообщение и возвращает его ЭЦП protected boolean engineVerify(byte[] sigBytes) – проверяет истинность подписанного сообщения, где sigBytes – блок подписи. Возвращает true – если ЭЦП верна, в ином случае - false. protected void engineSetParameter(AlgorithmParameterSpec params) – функция не применяется. Выдает UnsupportedOperationException. protected void engineSetParameter(String param, Object value) – функция не применяется. Выдает UnsupportedOperationException. protected Object engineGetParameter(String param) – функция не применяется. Выдает UnsupportedOperationException
Примеры	см. com.digt.trusted.jce.provider.test.GOST3410v12Test

GOST3410KeySpec

Соответствует	Интерфейс для GOST3410PrivateKeySpec и GOST3410PublicKeySpec

Пакет	com.digt.trusted.jce.spec
Класс	GOST3410KeySpec
Совместимость с JCE	Совместим
Описание класса	Реализует интерфейс KeySpec
Методы класса	protected GOST3410KeySpec(GOST3410ParameterSpec spec) – стандартный конструктор параметров. public GOST3410ParameterSpec getParams() – возвращает параметры
Примеры	см. com.digt.trusted.jce.spec.GOST3410PrivateKeySpec com.digt.trusted.jce.spec.GOST3410PublicKeySpec

GOST3410ParameterSpec

Соответствует	Параметры ключей классов GOST3410
Пакет	com.digt.trusted.jce.spec
Класс	GOST3410ParameterSpec
Совместимость с JCE	Совместим
Описание класса	Реализует интерфейс AlgorithmParameterSpec
Методы класса	public GOST 3410 ParameterSpec(String P1, String P2) – стандартный конструктор параметров. public String getP1() – возвращает первый параметр. public String getP2() – возвращает второй параметр
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410PrivateKey com.digt.trusted.jce.provider.JDKGOST3410PublicKey и др.

GOST3410PublicKeySpec

Соответствует	Открытый ключ с параметрами в классах GOST 3410
Пакет	com.digt.trusted.jce.spec

Класс	GOST3410PublicKeySpec
Совместимость с JCE	Совместим
Описание класса	Содержит в себе открытый ключ с параметрами. Расширяет класс GOST3410KeySpec
Методы класса	public GOST 3410 PublicKeySpec(byte[] y, GOST 3410 ParameterSpec spec) – инкапсулирует в объект ключ с параметрами. public byte[] getY() – возвращает ключ
Примеры	см. com.digt.trusted.jce.provider.JDKGOST3410PublicKey com.digt.trusted.jce.provider.JDKGOST3410ELPublicKey

CAPICertStore

Соответствует	Работа с хранилищами сертификатов, которые обслуживает ОС
Пакет	com.digt.trusted.jce.provider
Класс	CAPICertStore
Совместимость с JCE	Класс работает только с хранилищами сертификатов, расположенными и обслуживаемыми операционной системы
Описание класса	Получает список сертификатов и CRL из указанного хранилища сертификатов (в ОС) (CurrentUser, LocalComputer/MY, CA, ROOT, TRUST) с получением возможности дальнейшего поиска в нем сертификата (или CRL) по какому-нибудь параметру
Методы класса	public CAPICertStore() – пустой конструктор. protected CAPICertStore(CertStoreSpi storeSpi, Provider provider, String type, CertStoreParameters params, Collection certs, Collection crls) - конструктор создает объект типа CAPICertStore. Параметры: storeSpi – реализация провайдера provider – провайдер type – тип params – инициализирующие параметры (могут быть null) certs – устанавливаемые сертификаты crls – устанавливаемые списки отозванных сертификатов public final Collection getCertificates(CertSelector selector) – возвращает список сертификатов, соответствующих условию выбора selector.

	public final Collection getAllCertificates() – возвращает объекты всех сертификатов из хранилища. public final Collection getCRLs(CRLSelector selector) – возвращает список CRL, соответствующих условию выбора selector. public final Collection getAllCRLs() – возвращает объекты всех CRL из хранилища. public CAPICertStore getInstance(String type, CertStoreParameters params) – получает объект хранилища сертификатов вида CAPICertStore, где содержатся сертификаты полученные из системного хранилища сертификатов. Параметры: type – имя открываемого системного хранилища <расположение / место назначения> (например: "CurrentUser/My"). params – инициализирующие параметры (может быть null). public final CertStoreParameters getCertStoreParameters() – возвращает параметры. public final String getType() – возвращает тип. public final Provider getProvider() – возвращает провайдер. public static final String getDefaultType() – возвращает тип по умолчанию
Примеры	см. com.digt.trusted.jce.provider.test.CryptoProCSPKeyStoreTest

CryptoProCSPKeyStore

Соответствует	Работа с хранилищами ключей и сертификатов, которые обслуживает ОС
Пакет	com.digt.trusted.jce.provider
Класс	CryptoProCSPKeyStore
Совместимость с JCE	Класс работает только с хранилищем ключей, расположенным и обслуживаемым операционной системы, а не с java, где построена своя реализация хранилища ключей и сертификатов в едином пространстве
Описание класса	Хранит информацию, связанную с ключевыми контейнерами и сертификатами
Методы класса	public Enumeration engineAliases() – возвращает список хранимых элементов. public boolean engineContainsAlias(String alias) – дает

	ответ, содержит ли класс элемент с именем alias. public void engineDeleteEntry(String alias) – удаляет элемент по имени. public Certificate engineGetCertificate(String alias) – возвращает сертификат, соответствующий имени alias. public String engineGetCertificateAlias(Certificate cert) – по сертификату возвращает его alias имя. public Certificate[] engineGetCertificateChain(String alias) – возвращает цепочку сертификатов для элемента с именем alias. public Date engineGetCreationDate(String alias) – не используется. Всегда возвращает null. public Key engineGetKey(String alias, char[] password) – возвращает объект секретного ключа для данного элемента alias. Параметры: alias – имя ключевого контейнера. password – пароль на контейнер. При открытии контейнера CSP сформирует запрос на ввод pin-кода. public boolean engineIsCertificateEntry(String alias) – проверяет является ли alias содержащим сертификаты. public boolean engineIsKeyEntry(String alias) – дает ответ, хранит ли данный alias, ключ. public void engineSetCertificateEntry(String alias, Certificate cert) – устанавливает сертификат в системное хранилище сертфикатов. Параметры: alias – первоначально задается любой псевдоним. После установки сертификата в системное хранилище он будет иметь строковый псевдоним хеш-значения, вычисленного из содержимого сертификата. cert – устанавливаемый сертификат. public void engineSetKeyEntry(String alias, byte[] key, Certificate[] chain) – устанавливает ключ в хранилище, с соответствующей ему цепочкой сертификатов. Параметры: alias – имя элемента, с которым устанавливается ключ. key – устанавливаемый ключ. chain - цепочка сертификатов, где в chain[0] сертификат, содержащий открытый ключ соответствующей секретному key. public void engineSetKeyEntry (String alias, Key key, char[] password, Certificate[] chain) – устанавливает сертификат в хранилище сертификатов, если сертификат соответствует ключевому контейнеру. Параметры: alias – имя элемента, с которым устанавливается ключ. key – устанавливаемый ключ. password – задается пароль на доступ к хранилищу. chain - цепочка сертификатов, где в chain[0] сертификат, содержащий открытый ключ соответствующей секретному key. public int engineSize() – возвращает количество

	элементов хранилища. public void engineLoad (InputStream stream, char[] password) – загружает ключевые контейнеры со списками(цепочками) установленных в них сертификатов, и сертификаты из указанного в stream расположении. Параметры: stream – имя открываемого системного хранилища <расположение/место назначения> (например: "CurrentUser/My"). password – не используется (передается null). public void engineStore(OutputStream stream, char[] password) – метод не используется
Примеры	см. com.digt.trusted.jce.provider.test.CryptoProCSPKeyStoreTest

TSPHelper

Соответствует	Получение и проверка штампа времени
Пакет	com.digt.trusted.tsp
Класс	TSPHelper
Совместимость с JCE	Вспомогательный класс для получения/проверки штампа времени с локальной/удаленной службы штампов времени
Описание класса	Хранит информацию, связанную с ключевыми контейнерами и сертификатами
Методы класса	public TSPHelper(CMSProcessable msg) – конструктор создает экземпляр объекта, проинициализированный указанными в параметрах данными для последующего получения или проверки штампа времени. Параметры: msg - данные, для которых должен быть вычислен data imprint хэш, включаемый в запрос на получение штампа времени. public byte[] getDataImprint() - возвращает message imprint GOST3411 хэш для данных, переданных через конструктор или метод setDataImprint. public void setDataImprint(CMSProcessable msg) - генерирует ГОСТ3411 хэш указанных данных и сохраняет его для дальнейшего использования. Параметры: msg - данные, для которых должен быть вычислен

	data imprint хэш, включаемый в запрос на получение штампа времени. public static void validateCert(TimeStampToken tok) - проверяет валидность сертификата, содержащегося в штампе времени, и корректность подписи штампа времени на этом сертификате. Параметры: tok - проверяемый штамп времени public TimeStampToken getTimeStampLocal(PrivateKey key, X509Certificate ca, X509Certificate cert) - создает штамп времени с помощью встроенного в провайдер генератора. Возвращает штамп времени. Параметры: key - приватный ключ локального TSA ca - сертификат УЦ, на котором был сгенерирован сертификат локального TSA cert - сертификат локального TSA public TimeStampToken getTimeStampOnline(String tsaUrl) - запрашивает штамп времени с сервера служб штампов времени. Проверяет его на валидность и соответствие переданному запросу. Возвращает валидный штамп времени, который вернул сервер. Параметры: tsaUrl - адрес сервера служб штампов времени. public static AttributeTable convertToAttr(TimeStampToken tok) - возвращает штамп времени как экземпляр AttributeTable в формате, пригодном для использования в качестве аттрибута при генерации подписи. public TimeStampToken verifySigner(SignerInformation signer) - производит проверку штампа времени в указанном экземпляре SignerInformation. Если в данных подписчика штамп времени отсутствует, проверка считается успешной. Возвращает метку штампа времени или null в случае ее отсутствия. Параметры: signer - данные подписчика, в которых предполагается наличие аттрибута штампа времени
Примеры	см. com.digt.trusted.tsp.test.TSPTest

Техническая поддержка

По вопросам технической поддержки ПО Trusted Java обращайтесь: - по электронной почте: support@trusted.ru - по телефону: 8 (8362) 33-70-50 - по адресу: 424033, Россия, РМЭ, г. Йошкар-Ола, ул. Петрова, д.1, а/я 67.

Систематическое техническое сопровождение организаций по вопросам

работы с ПО Trusted Java осуществляется при условии оплаты стоимости годового пакета технических услуг. В течение этого срока разработчик бесплатно поставляет новые реализации продукта в рамках приобретенной версии. Новые версии продукта поставляются в соответствии с прайс-листом.

О компании-разработчике

Компания «Цифровые технологии» – российский разработчик и поставщик программного обеспечения в области защиты информации, систем электронного документооборота и хранения данных. Основной сферой деятельности компании является разработка, внедрение и поддержка криптографических продуктов и решений для государственных и коммерческих структур.

Телефон: (8362) 33-70-50 Интернет: http://www.trusted.ru E-mail: info@trusted.ru