



# Trusted Java 1.5.2

**Руководство разработчика**



**Цифровые технологии ®**  
2009 год

Настоящий документ содержит описание программного интерфейса доступа к методам программного продукта Trusted Java версии 1.5.2. В документе описываются объекты управления, дается характеристика провайдера JCE и его особенностей. Приведены параметры настройки провайдера JSSE, а также подробное описание классов, реализующих ГОСТ алгоритмы, и примеры их применения.

Документ предназначен для разработчиков ПО, прикладных и системных программистов, аналитиков, разрабатывающих приложения или модули интеграции внешних систем с программным продуктом Trusted Java.

По всем вопросам обращайтесь [в службу технической поддержки](#) или [на форум компании-разработчика](#)

## Новинки версии

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

<b>Объект</b>	<b>Описание</b>
<b>Шифрование и расшифрование</b>	<i>Шифрование</i> – это процесс выбора данных (текст-оригинал) и короткой последовательности символов (ключ), и вывода данных (шифрованный текст), являющихся бессмысленными для тех, кто не знает код. <i>Расшифрование</i> – это обратный процесс, то есть выбор зашифрованного текста и кодовой последовательности, и вывод текста-оригинала.
<b>Шифрование с использованием пароля (PBE)</b>	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».
<b>Шифр</b>	Шифрование и расшифрование осуществляется с использованием шифров. <i>Шифр</i> – это объект, способный выполнять шифрование и расшифрование в соответствии со схемой шифрования (алгоритмом).
<b>Согласование ключа</b>	<i>Согласование ключа</i> – это протокол, при помощи которого 2 или более пользователя могут установить одни криптографические ключи, не прибегая к обмену какой-либо секретной информацией.
<b>Код аутентификации сообщений</b>	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 хэш-функциях, 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.

## ЖСЕ и ее особенности

ЖСЕ 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.

ЖСЕ основана на тех же структурных принципах, что и другие программы ЖСА: 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, или IDEA;
- 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 Diffie-Hellman 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 HMAC-SHA1, 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

В этой главе приведено описание настроек провайдера 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 2.5 и выше, а так же 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\_java152.jar в каталог `$path_to_jre/lib/ext`
- 2) Указать, что при создании SSL соединений должен использоваться DigtSocketFactory класс в файле настроек безопасности `<jre>\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.3411 – основной класс GOST3411Digest;
- ГОСТ Р.28147-89 - основной класс GOST28147Engine;
- ГОСТ Р.3410-94 – класс генерации ключей GOST3410KeyPairGenerator, класс ЭЦП GOST3410Signer;
- ГОСТ Р.3410-2001 – класс генерации ключей GOST3410ELKeyPairGenerator, класс ЭЦП GOST3410 EL Signer.

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. 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 djcр-библиотеки 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.

### Классы, реализующие ГОСТ алгоритмы

[GOST28147Engine](#)

[GOST3410](#)

[GOST3410ELKeyPairGenerator](#)

[GOST3410Signer](#)

[GOSTDHKeyPairGenerator](#)

[GOSTECDHKeyPairGenerator](#)

[GOST3410KeyParameters](#)

[GOST3410KeyGenParameterSpec](#)

[GOST3410PrivateKeyParameters](#)

[GOST3410ELPrivateKeyParameters](#)



[GOST3410ELPublicKeyParameters](#)  
[GOST3410Parameter](#)  
[GOSTDHAgreement](#)  
[GOSTSessionKey](#)  
[GOST3410PrivateKey](#)  
[GOST3410PublicKey](#)  
[JDKGOST3410PrivateKey](#)  
[JDKGOST3410ELPrivateKey](#)  
[JDKGOST3410ELPublicKey](#)  
[JDKGOST3410Signer](#)  
[GOST3410KeySpec](#)  
[GOST3410ParameterSpec](#)  
[GOST3410PublicKeySpec](#)  
[CACertStore](#)  
[CryptoProCSPKeyStore](#)  
[TSPHelper](#)

## GOST28147Engine

<b>Соответствует</b>	ГОСТ 28147-89
<b>Пакет</b>	com.digt.trusted.crypto.engines
<b>Класс</b>	GOST28147Engine
<b>Совместимость с JCE</b>	<ul style="list-style-type: none"> <li>● Имена методов идентичны другим блочным шифрам jce.</li> <li>● Добавлены методы для установки режима шифрования и вектора шифрования. Добавлен метод для инициализации шифра с сессионным ключом.</li> <li>● Отличным по логике от шифров jce выступает метод <b>processBlock</b>, которому на вход подается все сообщение (а не часть блока), 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. <a href="#">примеры использования класса</a>). Padding установлен в RANDOM_PADDING, поэтому нет смысла выбирать другой заполнитель, в java везде используется тип “/NoPadding”</li> </ul>
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>● Шифрование по алгоритму ГОСТ 28147-89</li> <li>● Реализует интерфейс <b>BlockCipher</b></li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>● public void init(boolean encrypting, CipherParameters params) – инициализирует шифр ГОСТ 28147-89. <i>Параметры:</i> 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).</li> <li>● public void init(boolean forEncryption, GOSTSessionKey seskey) – инициализирует шифр ГОСТ 28147-89. <i>Параметры:</i> encrypting – если «true» - шифр инициализируется для зашифрования,</li> </ul>

	<p>если «false» - для расшифрования.  seskey – сессионный ключ формата GOSTSessionKey.</p> <ul style="list-style-type: none"> <li>public int processBlock (byte[] in, int inOff, byte[] out, int outOff) – выполняет шифрование входного блока и выводит результат в выходной массив.  <i>Параметры:</i>  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.</li> <li>public int processBlock (byte[] in, int inOff, int length, byte[] out, int outOff) – выполняет шифрование входного блока и выводит результат в выходной массив.  Метод добавлен для совместимости с классами режимов и просто выполняет вызов public int processBlock (byte[] in, int inOff, byte[] out, int outOff).  <i>Параметры:</i>  len – не используется.</li> </ul> <p>Остальные параметры см. public int processBlock(byte[] in, int inOff, byte[] out, int outOff).</p> <ul style="list-style-type: none"> <li>public String getAlgorithmName() – возвращает имя алгоритма блочного шифра.</li> <li>public void setMode (String mode) – устанавливает режим шифрования (используется для установки блочного режима “ECB”).</li> <li>public void setMode (String mode, byte[] IV) – устанавливает режим шифрования (используется для установки потоковых режимов “CFB” и “OFB”) и начального вектора инициализации.</li> <li>public String getMode() – возвращает используемый режим шифрования.</li> <li>public int getBlockSize() – возвращает размер одного блока при шифровании в ГОСТ 28147-89 (количество байт)</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.test.GOST28147Test

## GOST3410

<b>Соответствует</b>	интерфейс подписи ГОСТ 3410
<b>Пакет</b>	com.digt.trusted.crypto
<b>Класс</b>	GOST3410
<b>Совместимость с JCE</b>	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.
<b>Описание класса</b>	Интерфейс описывает общую структуру классов подписи ГОСТ 3410
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>public void init (boolean forSigning, CipherParameters param) – инициализирует класс подписчика для генерации подписи или проверки подписи.  <i>Параметры:</i>  forSigning – если «true» - генерация подписи, в ином случае - проверка.  param – ключевые параметры для подписи (это закрытый или открытый ключ).</li> </ul>

	<ul style="list-style-type: none"> <li>public byte[] generateSignature(byte[] message) – подписывает переданное сообщение. <i>Параметры:</i> message – сообщение для подписи. Возвращает блок данных, являющийся подписанным сообщением (или просто ЭЦП).</li> <li>public boolean verifySignature (byte[] message, byte[] sign) – проверяет истинность подписанного сообщения. <i>Параметры:</i> message – сообщение, которое было подписано. sign – значение подписи.</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.crypto.signers.GOST3410Signer com.digt.trusted.crypto.signers.GOST3410ELSigner com.digt.trusted.jce.provider.JDKGOST3410Signer com.digt.trusted.jce.provider.JDKGOST3410ELSigner

## GOST3410ELKeyPairGenerator

<b>Соответствует</b>	Генератор ключей по ГОСТ 3410-2001
<b>Пакет</b>	com.digt.trusted.crypto.generators
<b>Класс</b>	GOST3410ELKeyPairGenerator
<b>Совместимость с JCE</b>	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)
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>Производится ключевая пара согласно ГОСТ Р 34.10-94, сохраняемая в ключевом контейнере.</li> <li>Предназначена для ЭЦП.</li> <li>Реализует интерфейс AsymmetricCipherKeyPairGenerator.</li> <li>Класс GOST 3410 ELKeyPairGenerator сходен с GOST 3410 KeyPairGenerator за исключением возвращаемого открытого ключа (в 512 бит)</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>public void init (KeyGenerationParameters param) – задает начальные параметры генератора. <i>Параметры:</i> param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара.</li> <li>public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару по ГОСТ 3410-2001, открытый ключ длиной 512 бит. Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.test.GOST3410ELTest

## GOST3410Signer

<b>Соответствует</b>	Электронная цифровая подпись ГОСТ 3410-94
<b>Пакет</b>	com.digt.trusted.crypto.signers
<b>Класс</b>	GOST3410Signer



<b>Совместимость с JCE</b>	СОВМЕСТИМ
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Подпись сообщения и проверка подписи по ГОСТ 3410-94.</li> <li>■ Реализует интерфейс GOST3410.</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ <code>public void init (boolean forSigning, CipherParameters param)</code> – инициализирует класс подписчика для генерации подписи или проверки подписи. <i>Параметры:</i> <code>forSigning</code> – если «true» - генерация подписи, в противном случае - проверка. <code>param</code> – ключевые параметры для подписи (закрытый или открытый ключ).</li> <li>■ <code>public byte[] generateSignature(byte[] message)</code> – подписывает переданное сообщение. <i>Параметры:</i> <code>message</code> – хеш сообщения для подписи. Возвращает подписанное сообщение.</li> <li>■ <code>public boolean verifySignature (byte[] message, byte[] sign)</code> – проверяет истинность подписанного сообщения. <i>Параметры:</i> <code>message</code> – хеш сообщения, которое было подписано. <code>sign</code> – значение подписи</li> </ul>
<b>Примеры</b>	<p>см.</p> <p><code>com.digt.trusted.jce.provider.test.GOST3410Test</code>  <code>com.digt.trusted.jce.provider.test.CertTest</code> – проверка подписи сертификата</p>

## GOSTDHKeyPairGenerator

<b>Соответствует</b>	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-94
<b>Пакет</b>	<code>com.digt.trusted.crypto.generators</code>
<b>Класс</b>	<code>GOSTDHKeyPairGenerator</code>
<b>Совместимость с JCE</b>	Аналогия имен методов и структуры класса. Так как все параметры заданы в КриптоПро CSP, то <code>GOSTDHKeyPairGenerator</code> 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)
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Производится ключевая пара согласно ГОСТ Р 34.10-94, сохраняемая в ключевом контейнере.</li> <li>■ Она предназначена для обмена сессионными ключами и ЭЦП.</li> <li>■ Реализует интерфейс <code>AsymmetricCipherKeyPairGenerator</code></li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ <code>public void init (KeyGenerationParameters param)</code> – задает начальные параметры генератора. <i>Параметры:</i> <code>param</code> – параметр, в котором передается имя контейнера, для которого будет генерироваться ключевая пара.</li> <li>■ <code>public AsymmetricCipherKeyPair generateKeyPair()</code> – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-94, открытый ключ длиной 1024 бит. Возвращает <code>AsymmetricCipherKeyPair</code> – содержит сгенерированные ключи</li> </ul>

## GOSTECDHKeyPairGenerator

<b>Соответствует</b>	Генератор ключей для алгоритма Diffie-Hellman согласно ГОСТ 3410-2001
<b>Пакет</b>	com.digt.trusted.crypto.generators
<b>Класс</b>	GOSTECDHKeyPairGenerator
<b>Совместимость с JCE</b>	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)
<b>Описание класса</b>	<ul style="list-style-type: none"><li>● Производится ключевая пара согласно ГОСТ Р 34.10-2001, сохраняемая в ключевом контейнере.</li><li>● Она предназначена для обмена сессионными ключами и ЭЦП.</li><li>● Реализует интерфейс AsymmetricCipherKeyPairGenerator</li></ul>
<b>Методы класса</b>	<ul style="list-style-type: none"><li>● public void init (KeyGenerationParameters param) – задает начальные параметры генератора. <i>Параметры:</i> param – параметр, где передается имя контейнера, для которого будет генерироваться ключевая пара.</li><li>● public AsymmetricCipherKeyPair generateKeyPair() – генерирует ключевую пару для обмена сессионными ключами и ЭЦП по ГОСТ 3410-2001, открытый ключ длиной 512 бит. Возвращает AsymmetricCipherKeyPair – содержит сгенерированные ключи</li></ul>

## GOST3410KeyParameters

<b>Соответствует</b>	Ключевые параметры для алгоритмов ГОСТ 3410
<b>Пакет</b>	com.digt.trusted.crypto.params
<b>Класс</b>	GOST3410KeyParameters
<b>Совместимость с JCE</b>	Аналогичен другим подобным классам
<b>Описание класса</b>	<ul style="list-style-type: none"><li>● Служит контейнером для ключевых параметров из КриптоПро CSP.</li><li>● Общий класс в классах GOST 3410, GOST 3410 EL, GOSTDH.</li><li>● Расширяет класс AsymmetricKeyParameter</li></ul>
<b>Методы класса</b>	<ul style="list-style-type: none"><li>● protected GOST3410KeyParameters(boolean isPrivate, GOST3410Parameters params) – стандартный конструктор, создает класс с параметрами. <i>Параметры:</i> isPrivate – устанавливает флаг типа параметров ключа. params – ключевые параметры.</li><li>● public GOST 3410 Parameters getParameters() – возвращает параметры класса.</li><li>● public boolean equals(Object obj) – сравнивает 2 объекта GOST3410KeyParameters</li></ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.crypto.params.GOST3410PublicKeyParameters com.digt.trusted.crypto.signers.GOST3410Signer com.digt.trusted.crypto.signers.GOST3410ELSigner и др.

## GOST3410KeyGenParameterSpec

<b>Соответствует</b>	Параметр при генерации ключевых пар ГОСТ 3410
<b>Пакет</b>	com.digt.trusted.crypto.params
<b>Класс</b>	GOST3410KeyGenParameterSpec
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"><li>Хранит имя ключевого контейнера, в котором хранится ключевая пара.</li><li>Реализует интерфейс AlgorithmParameterSpec</li></ul>
<b>Методы класса</b>	public GOST3410KeyGenParameterSpec(String container) – стандартный конструктор <i>Параметры:</i> container – имя контейнера, в котором находится ключевая пара. public String getContainer() – возвращает имя контейнера
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.test.GOST3410Test com.digt.trusted.jce.provider.test.GOST3410ELTest

## GOST3410PrivateKeyParameters

<b>Соответствует</b>	Закрытый ключ с параметрами ГОСТ 3410-94
<b>Пакет</b>	com.digt.trusted.crypto.params
<b>Класс</b>	GOST3410PrivateKeyParameters
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"><li>Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами.</li><li>Расширяет класс GOST3410KeyParameters</li></ul>
<b>Методы класса</b>	<ul style="list-style-type: none"><li>public GOST3410PrivateKeyParameters(String x, GOST3410Parameters params) – стандартный конструктор. <i>Параметры:</i> x – имя контейнера, в котором находится ключевая пара. params – параметры к ключу.</li><li>public String getX() – возвращает закрытый ключ (имя контейнера, в котором он лежит).</li><li>public boolean equals(Object obj) – сравнивает 2 объекта GOST3410PrivateKeyParameters</li></ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.crypto.generators.GOST3410KeyPairGenerator com.digt.trusted.crypto.signers.GOST3410Signer com.digt.trusted.jce.provider.JDKGOST3410PrivateKey и др

## GOST3410ELPrivateKeyParameters

<b>Соответствует</b>	Закрытый ключ с параметрами ГОСТ 3410-2001
<b>Пакет</b>	com.digt.trusted.crypto.params
<b>Класс</b>	GOST3410ELPrivateKeyParameters
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"><li>Хранит закрытый ключ (имя контейнера, в котором хранится ключевая пара) с дополнительными параметрами.</li><li>Расширяет класс GOST3410KeyParameters</li></ul>
<b>Методы класса</b>	<ul style="list-style-type: none"><li>public GOST3410ELPrivateKeyParameters(String x, GOST3410Parameters</li></ul>

	<p>params) – стандартный конструктор.  <i>Параметры:</i>  x – имя контейнера, в котором находится ключевая пара.  params – параметры к ключу.</p> <ul style="list-style-type: none"> <li>■ public String getX () – возвращает закрытый ключ (имя контейнера, в котором он хранится).</li> <li>■ public boolean equals(Object obj) – сравнивает 2 объекта GOST3410ELPrivateKeyParameters</li> </ul>
<b>Примеры</b>	<p><i>см.</i>  com.digt.trusted.crypto.generators.GOST3410ELKeyPairGenerator  com.digt.trusted.crypto.signers.GOST3410ELSigner  com.digt.trusted.jce.provider.JDKGOST3410ELPrivateKey и др.</p>

## GOST3410ELPublicKeyParameters

<b>Соответствует</b>	Открытый ключ с параметрами ГОСТ 3410-2001
<b>Пакет</b>	com.digt.trusted.crypto.params
<b>Класс</b>	GOST3410ELPublicKeyParameters
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Хранит открытый ключ с дополнительными параметрами.</li> <li>■ Расширяет класс GOST3410KeyParameters</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ public GOST 3410 ELPublicKeyParameters (byte[] y, GOST 3410 Parameters params) – конструктор создает открытый ключ с параметрами.</li> </ul> <p><i>Параметры:</i>  y – открытый ключ в виде блока байт.  params – параметры открытого ключа.</p> <ul style="list-style-type: none"> <li>■ public byte[] getY() – возвращает открытый ключ (без параметров)</li> </ul>
<b>Примеры</b>	<p><i>см.</i>  com.digt.trusted.crypto.generators.GOST3410ELKeyPairGenerator  com.digt.trusted.crypto.signers.GOST3410ELSigner  com.digt.trusted.jce.provider.JDKGOST3410ELPublicKey и др.</p>

## GOST3410Parameter

<b>Соответствует</b>	Параметры открытого ключа в asn 1 по ГОСТ3410-94
<b>Пакет</b>	com.digt.trusted.asn1.x509
<b>Класс</b>	GOST3410Parameter
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Служит для хранения параметров открытого ключа вида asn 1, передается в SubjectPublicKeyInfo в классе GOST3410Parameter</li> <li>■ Реализует интерфейс DEREncodable</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ public static GOST3410Parameter getInstance(ASN1TaggedObject obj, boolean explicit) – no comments.</li> <li>■ public static GOST3410Parameter getInstance(Object obj) – no comments.</li> <li>■ public GOST3410Parameter(String p1, String p2) – конструктор с двумя параметрами на входе.</li> <li>■ public GOST3410Parameter(ASN1Sequence seq) – no comments.</li> <li>■ public String getP1() – возвращает первый параметр.</li> </ul>

	<ul style="list-style-type: none"> <li>■ <code>public String getP2()</code> – возвращает второй параметр.</li> <li>■ <code>public DERObject getDERObject()</code> – возвращает объект параметров в <code>DERObject</code></li> </ul>
<b>Примеры</b>	<i>см.</i> <code>com.digt.trusted.jce.provider.JDKGOST3410PublicKey</code>

## GOSTDHAgreement

<b>Соответствует</b>	Основа для алгоритма Diffie-Hellman
<b>Пакет</b>	<code>com.digt.trusted.crypto.agreement</code>
<b>Класс</b>	<code>GOSTDHAgreement</code>
<b>Совместимость с JCE</b>	Присутствуют те же методы, что и в классах той же группы ( <code>Agreement</code> ). Добавлены новые методы с новыми функционалами. Класс не вычисляет общий закрытый ключ, в нем происходит экспорт (импорт) сессионного ключа зашифрованного на общем ключе обмена
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ В классе сохраняется собственный закрытый ключ (<code>init</code>)</li> <li>■ Помещается открытый ключ отправителя (<code>calculateAgreement</code>)</li> <li>■ Экспортируется сессионный ключ (<code>exportSessionKey</code>)</li> <li>■ Импортируется сессионный ключ (<code>importSessionKey</code>)</li> </ul> Подробности ниже в описании методов
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ <code>public void init(CipherParameters param)</code> – сохраняется в классе собственный закрытый ключ, переданный в классе параметров (имя ключевого контейнера).</li> <li>■ <code>public void init(Key key)</code> – сохраняется закрытый ключ, переданный в классе <code>privatekey</code>.</li> <li>■ <code>public BigInteger calculateAgreement(GOST3410PublicKeyParameters pub)</code> – сохраняет. Возвращает всегда <code>null</code>. (<code>BigInteger</code> – в целях совместимости).</li> <li>■ <code>public CipherParameters getPrivateKeyParameters()</code> – возвращает параметры закрытого ключа. Функция неактуальна, оставлена в целях совместимости.</li> <li>■ <code>public GOST3410PublicKeyParameters getPublicKeyParameters()</code> – возвращает открытый ключ с параметрами.</li> <li>■ <code>public String getContainer()</code> – возвращает имя контейнера.</li> <li>■ <code>public byte[] exportSessionKey(GOSTSessionKey gsk)</code> – экспортирует сессионный ключ.  <i>Параметры:</i>  <code>gsk</code> – структура сессионного ключа на экспорт            Возвращает блок данных, готовых к передаче на сторону получателя.</li> <li>■ <code>public GOSTSessionKey importSessionKey(byte[] impkey)</code> – импортирует сессионный ключ.  <i>Параметры:</i>  <code>impkey</code> – принятый блок сессионного ключа.            Возвращает структуру восстановленного сессионного ключа</li> </ul>

## GOSTSessionKey

<b>Соответствует</b>	Сессионный ключ ГОСТ 28147
<b>Пакет</b>	<code>com.digt.trusted.crypto.GOSTSessionKey</code>
<b>Класс</b>	<code>GOSTSessionKey</code>
<b>Совместимость с JCE</b>	<code>no comments</code>



<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Класс создан для хранения, генерации и передачи сессионного ключа (с параметрами).</li> <li>■ Реализует интерфейс <code>SecretKey</code></li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ <code>public GOSTSessionKey()</code> – пустой конструктор.</li> <li>■ <code>public GOSTSessionKey (byte[] EncodedObject)</code> – конструктор, создающий из пришедшего блока <code>EncodedObject</code> полноценный класс (импортирует блок в структуру полей класса).</li> <li>■ <code>public byte[] getEncoded()</code> – кодирует поля класса с возможностью дальнейшего экспорта класса.</li> <li>■ <code>public void genKey (String containerName)</code> – генерирует сессионный ключ, шифруемый ключом обмена.</li> <li>■ <code>public void setkeyBlob (byte[] keyBlob)</code> – устанавливает блок, содержащий шифрованный сессионный ключ.</li> <li>■ <code>public void setKP_IV (byte[] KP_IV)</code> – устанавливает начальный вектор инициализации.</li> <li>■ <code>public void setKP_PADDING (byte[] KP_PADDING)</code> – устанавливает умалчиваемый режим заполнения.</li> <li>■ <code>public void setKP_MODE (byte[] KP_MODE)</code> – устанавливает умалчиваемый режим шифрования.</li> <li>■ <code>public void sethProv (int hProv)</code> – устанавливает хендл провайдера.</li> <li>■ <code>public void sethKey(int hKey)</code> – устанавливает хендл ключа.</li> <li>■ <code>public byte[] getkeyBlob()</code> – возвращает блок, содержащий шифрованный сессионный ключ.</li> <li>■ <code>public byte[] getKP_IV()</code> – возвращает начальный вектор инициализации.</li> <li>■ <code>public byte[] getKP_PADDING()</code> – возвращает умалчиваемый режим заполнения.</li> <li>■ <code>public byte[] getKP_MODE()</code> – возвращает умалчиваемый режим шифрования.</li> <li>■ <code>public int gethProv()</code> – возвращает хендл провайдера.</li> <li>■ <code>public int gethKey()</code> – возвращает хендл ключа.</li> <li>■ <code>public String getAlgorithm()</code> – no comments.</li> <li>■ <code>public String getFormat()</code> – no comments.</li> <li>■ <code>public void destroy()</code> – см. <a href="#">«Некоторые общие методы»</a></li> </ul>
<b>Примеры</b>	<p>см.  <code>com.digt.trusted.crypto.agreement.GOSTDHAgreement</code></p>

## GOST3410PrivateKey

<b>Соответствует</b>	Общий интерфейс для закрытых ключей классов GOST3410
<b>Пакет</b>	<code>com.digt.trusted.jce.interfaces</code>
<b>Класс</b>	<code>GOST3410PrivateKey</code>
<b>Совместимость с JCE</b>	no comments
<b>Описание класса</b>	Расширяет интерфейс <code>PrivateKey</code>
<b>Методы класса</b>	<code>public String getX ()</code> – возвращает ссылку на закрытый ключ

## GOST3410PublicKey

<b>Соответствует</b>	Общий интерфейс для открытых ключей классов GOST3410
<b>Пакет</b>	<code>com.digt.trusted.jce.interfaces</code>
<b>Класс</b>	<code>GOST3410PublicKey</code>
<b>Совместимость с JCE</b>	no comments
<b>Описание класса</b>	Расширяет интерфейсы <code>GOST3410Key</code> , <code>PublicKey</code>

<b>Методы класса</b>	public byte[] getY() – возвращает открытый ключ
----------------------	---

## JDKGOST3410PrivateKey

<b>Соответствует</b>	Закрытый ключ по ГОСТ 3410-94 на уровне криптопровайдера
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	JDKGOST3410PrivateKey
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером.</li> <li>■ Реализует интерфейс GOST3410PrivateKey</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ public JDKGOST 3410 PrivateKey(GOST 3410 PrivateKey key) – инкапсулирует закрытый ключ типа GOST 3410 PrivateKey в объект.</li> <li>■ public JDKGOST 3410 PrivateKey(GOST 3410 PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST 3410 PrivateKeySpec в объект.</li> <li>■ public JDKGOST3410PrivateKey(GOST3410PrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeyParameters в объект.</li> <li>■ public String getAlgorithm() – возвращает алгоритм.</li> <li>■ public String getFormat() – возвращает тип хранения и передачи ключа.</li> <li>■ public String getX() – возвращает закрытый ключ.</li> <li>■ public byte[] getEncoded() – возвращает всегда null</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.JDKGOST3410Signer com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410 com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH

## JDKGOST3410ELPrivateKey

<b>Соответствует</b>	Закрытый ключ по ГОСТ 3410-2001 на уровне криптопровайдера
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	JDKGOST3410ELPrivateKey
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ Хранит в классе закрытый ключ, используется во взаимодействии с криптопровайдером.</li> <li>■ Реализует интерфейс GOST3410PrivateKey</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ public JDKGOST 3410 ELPrivateKey(GOST3410PrivateKey key) – инкапсулирует закрытый ключ типа GOST3410 PrivateKey в объект.</li> <li>■ public JDKGOST 3410 ELPrivateKey(GOST3410PrivateKeySpec key) – инкапсулирует закрытый ключ типа GOST3410PrivateKeySpec в объект.</li> <li>■ public JDKGOST 3410 ELPrivateKey(GOST3410ELPrivateKeyParameters key) – инкапсулирует закрытый ключ типа GOST3410ELPrivateKeyParameters в объект.</li> <li>■ public String getAlgorithm() – возвращает алгоритм.</li> <li>■ public String getFormat() – возвращает тип хранения и передачи ключа.</li> <li>■ public String getX () – возвращает закрытый ключ.</li> <li>■ public byte[] getEncoded() – возвращает всегда null</li> </ul>

<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.JDKGOST3410ELSigner com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410EL com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH
----------------	---

## JDKGOST3410ELPublicKey

<b>Соответствует</b>	Открытый ключ по ГОСТ 3410-2001 на уровне криптопровайдера
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	JDKGOST3410ELPublicKey
<b>Совместимость с JCE</b>	Без существенных отличий
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>● Хранит в классе открытый ключ с параметрами, используется во взаимодействии с криптопровайдером.</li> <li>● Реализует интерфейс GOST3410PublicKey</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>● public JDKGOST 3410 ELPublicKey(GOST 3410 PublicKey key) – инкапсулирует открытый ключ типа GOST 3410 PublicKey в объект.</li> <li>● public JDKGOST 3410 ELPublicKey(GOST 3410 PublicKeySpec spec) – инкапсулирует открытый ключ типа GOST 3410 PublicKeySpec в объект.</li> <li>● public JDKGOST 3410 ELPublicKey(GOST 3410 ELPublicKeyParameters params) – инкапсулирует открытый ключ типа GOST 3410 ELPublicKeyParameters в объект.</li> <li>● public JDKGOST 3410 ELPublicKey(SubjectPublicKeyInfo info) – инкапсулирует открытый ключ типа SubjectPublicKeyInfo в объект.</li> <li>● public String getAlgorithm() – возвращает алгоритм.</li> <li>● public String getFormat() – возвращает тип хранения и передачи ключа.</li> <li>● public byte[] getY() – возвращает открытый ключ.</li> <li>● public byte[] getEncoded() – декодирует объект в блок байт.</li> <li>● public GOST3410ParameterSpec getParams() – возвращает параметры ключа.</li> <li>● public String toString() – no comments</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.JDKGOST3410ELSigner com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOST3410EL com.digt.trusted.jce.provider.JDKKeyPairGenerator.GOSTDH

## JDKGOST3410Signer

<b>Соответствует</b>	Электронная цифровая подпись ГОСТ 3410-94 на уровне криптопровайдера
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	JDKGOST3410Signer
<b>Совместимость с JCE</b>	Совместим
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>● Подпись сообщения и проверка подписи по ГОСТ 3410-94.</li> <li>● Расширяет класс Signature</li> </ul>
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>● protected JDKGOST3410Signer(String name, Digest digest, GOST3410signer) – инкапсуляция определяющих параметров при цифровой подписи.</li> </ul>

	<p><b>Параметры:</b>  name – константное имя « GOST3411withGOST3410»  digest – класс, использующийся для получения хеша сообщения перед подписью.  signer – класс подписчика сообщений(GOST3410Signer).</p> <ul style="list-style-type: none"> <li>■ protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки ЭЦП. В параметре передается открытый ключ.</li> <li>■ protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи.</li> <li>■ protected void engineInitVerify(PublicKey publicKey) – инициализирует класс для проверки подписи. В параметре передается открытый ключ.</li> <li>■ protected void engineInitSign(PrivateKey privateKey, SecureRandom random) – инициализирует класс для подписи.</li> </ul> <p><b>Параметры:</b>  privateKey – передается закрытый ключ.  random – не используется (передается null).</p> <ul style="list-style-type: none"> <li>■ protected void engineInitSign (PrivateKey privateKey) – инициализирует класс для проверки ЭЦП. В параметре передается закрытый ключ.</li> <li>■ protected void engineUpdate (byte b) – «сохраняет в класс» байт подписанного сообщения.</li> <li>■ protected void engineUpdate (byte[] b, int off, int len) – «сохраняет в класс» блок байт подписанного сообщения. Начиная с off длиной len.</li> <li>■ protected byte[] engineSign() – подписывает сообщение и возвращает его ЭЦП.</li> <li>■ protected boolean engineVerify(byte[] sigBytes) – проверяет истинность подписанного сообщения, где sigBytes – блок подписи. Возвращает true – если ЭЦП верна, в ином случае - false.</li> <li>■ protected void engineSetParameter(AlgorithmParameterSpec params) – функция не применяется. Выдает UnsupportedOperationException.</li> <li>■ protected void engineSetParameter(String param, Object value) – функция не применяется. Выдает UnsupportedOperationException.</li> <li>■ protected Object engineGetParameter(String param) – функция не применяется. Выдает UnsupportedOperationException</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.test.GOST3410Test

## GOST3410KeySpec

<b>Соответствует</b>	Интерфейс для GOST3410PrivateKeySpec и GOST3410PublicKeySpec
<b>Пакет</b>	com.digt.trusted.jce.spec
<b>Класс</b>	GOST3410KeySpec
<b>Совместимость с JCE</b>	no comments
<b>Описание класса</b>	Реализует интерфейс KeySpec
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ protected GOST 3410 KeySpec(GOST 3410 ParameterSpec spec) – стандартный конструктор параметров.</li> <li>■ public GOST3410ParameterSpec getParams() – возвращает параметры</li> </ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.spec.GOST3410PrivateKeySpec com.digt.trusted.jce.spec.GOST3410PublicKeySpec

## GOST3410ParameterSpec

<b>Соответствует</b>	Параметры ключей классов GOST3410
<b>Пакет</b>	com.digt.trusted.jce.spec
<b>Класс</b>	GOST3410ParameterSpec
<b>Совместимость с JCE</b>	no comments
<b>Описание класса</b>	Реализует интерфейс AlgorithmParameterSpec
<b>Методы класса</b>	<ul style="list-style-type: none"><li>public GOST 3410 ParameterSpec(String P1, String P2) – стандартный конструктор параметров.</li><li>public String getP1() – возвращает первый параметр.</li><li>public String getP2() – возвращает второй параметр</li></ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.JDKGOST3410PrivateKey com.digt.trusted.jce.provider.JDKGOST3410PublicKey и др.

## GOST3410PublicKeySpec

<b>Соответствует</b>	Открытый ключ с параметрами в классах GOST 3410
<b>Пакет</b>	com.digt.trusted.jce.spec
<b>Класс</b>	GOST3410PublicKeySpec
<b>Совместимость с JCE</b>	no comments
<b>Описание класса</b>	<ul style="list-style-type: none"><li>Содержит в себе открытый ключ с параметрами.</li><li>Расширяет класс GOST3410KeySpec</li></ul>
<b>Методы класса</b>	<ul style="list-style-type: none"><li>public GOST 3410 PublicKeySpec(byte[] y, GOST 3410 ParameterSpec spec) – инкапсулирует в объект ключ с параметрами.</li><li>public byte[] getY() – возвращает ключ</li></ul>
<b>Примеры</b>	<i>см.</i> com.digt.trusted.jce.provider.JDKGOST3410PublicKey com.digt.trusted.jce.provider.JDKGOST3410ELPublicKey

## CAPICertStore

<b>Соответствует</b>	Работа с хранилищами сертификатов, которые обслуживает ОС
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	CAPICertStore
<b>Совместимость с JCE</b>	Класс работает только с хранилищами сертификатов, расположенными и обслуживаемыми операционной системы
<b>Описание класса</b>	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
<b>Методы класса</b>	<ul style="list-style-type: none"><li>public CAPICertStore() – пустой конструктор.</li><li>protected CAPICertStore(CertStoreSpi storeSpi, Provider provider, String type, CertStoreParameters params, Collection certs, Collection crls) - конструктор создает объект типа CAPICertStore.</li></ul> <p><i>Параметры:</i> storeSpi – реализация провайдера provider – провайдер</p>

	<p>type – тип  params – инициализирующие параметры (могут быть null)  certs – устанавливаемые сертификаты  crls – устанавливаемые списки отозванных сертификатов</p> <ul style="list-style-type: none"> <li>■ public final Collection getCertificates(CertSelector selector) – возвращает список сертификатов, соответствующих условию выбора selector.</li> <li>■ public final Collection getAllCertificates() – возвращает объекты всех сертификатов из хранилища.</li> <li>■ public final Collection getCRLs(CRLSelector selector) – возвращает список CRL, соответствующих условию выбора selector.</li> <li>■ public final Collection getAllCRLs() – возвращает объекты всех CRL из хранилища.</li> <li>■ public CAPICertStore getInstance(String type, CertStoreParameters params) – получает объект хранилища сертификатов вида CAPICertStore, где содержатся сертификаты полученные из системного хранилища сертификатов.</li> </ul> <p><b>Параметры:</b>  type – имя открываемого системного хранилища &lt;расположение / место назначения&gt; (например: "CurrentUser/My").  params – инициализирующие параметры (может быть null).</p> <ul style="list-style-type: none"> <li>■ public final CertStoreParameters getCertStoreParameters() – возвращает параметры.</li> <li>■ public final String getType() – возвращает тип.</li> <li>■ public final Provider getProvider() – возвращает провайдер.</li> <li>■ public static final String getDefaultType() – возвращает тип по умолчанию</li> </ul>
<b>Примеры</b>	<p>см.  com.digt.trusted.jce.provider.test.CryptoProCSPKeyStoreTest</p>

## CryptoProCSPKeyStore

<b>Соответствует</b>	Работа с хранилищами ключей и сертификатов, которые обслуживает ОС
<b>Пакет</b>	com.digt.trusted.jce.provider
<b>Класс</b>	CryptoProCSPKeyStore
<b>Совместимость с JCE</b>	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
<b>Описание класса</b>	Хранит информацию, связанную с ключевыми контейнерами и сертификатами
<b>Методы класса</b>	<ul style="list-style-type: none"> <li>■ public Enumeration engineAliases() – возвращает список хранимых элементов.</li> <li>■ public boolean engineContainsAlias(String alias) – дает ответ, содержит ли класс элемент с именем alias.</li> <li>■ public void engineDeleteEntry(String alias) – удаляет элемент по имени.</li> <li>■ public Certificate engineGetCertificate(String alias) – возвращает сертификат, соответствующий имени alias.</li> <li>■ public String engineGetCertificateAlias(Certificate cert) – по сертификату возвращает его alias имя.</li> <li>■ public Certificate[] engineGetCertificateChain(String alias) – возвращает цепочку сертификатов для элемента с именем alias.</li> <li>■ public Date engineGetCreationDate(String alias) – не используется. Всегда возвращает null.</li> <li>■ public Key engineGetKey(String alias, char[] password) – возвращает объект секретного ключа для данного элемента alias.</li> </ul>

	<p><i>Параметры:</i>  alias – имя ключевого контейнера.  password – пароль на контейнер. При открытии контейнера CSP сформирует запрос на ввод pin-кода.</p> <ul style="list-style-type: none"> <li>■ public boolean engineIsCertificateEntry(String alias) – проверяет является ли alias содержащим сертификаты.</li> <li>■ public boolean engineIsKeyEntry(String alias) – дает ответ, хранит ли данный alias, ключ.</li> <li>■ public void engineSetCertificateEntry(String alias, Certificate cert) – устанавливает сертификат в системное хранилище сертификатов.</li> </ul> <p><i>Параметры:</i>  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.</p> <ul style="list-style-type: none"> <li>■ public void engineSetKeyEntry(String alias, byte[] key, Certificate[] chain) – устанавливает ключ в хранилище, с соответствующей ему цепочкой сертификатов.</li> </ul> <p><i>Параметры:</i>  alias – имя элемента, с которым устанавливается ключ.  key – устанавливаемый ключ.  chain - цепочка сертификатов, где в chain[0] сертификат, содержащий открытый ключ соответствующей секретному key.</p> <ul style="list-style-type: none"> <li>■ public void engineSetKeyEntry (String alias, Key key, char[] password, Certificate[] chain) – устанавливает сертификат в хранилище сертификатов, если сертификат соответствует ключевому контейнеру.</li> </ul> <p><i>Параметры:</i>  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.</p> <ul style="list-style-type: none"> <li>■ public int engineSize() – возвращает количество элементов хранилища.</li> <li>■ public void engineLoad (InputStream stream, char[] password) – загружает ключевые контейнеры со списками(цепочками) установленных в них сертификатов, и сертификаты из указанного в stream расположении.</li> </ul> <p><i>Параметры:</i>  stream – имя открываемого системного хранилища &lt;расположение/место назначения&gt; (например: "CurrentUser/My").  password – не используется (передается null).</p> <ul style="list-style-type: none"> <li>■ public void engineStore(OutputStream stream, char[] password) – метод не используется</li> </ul>
<b>Примеры</b>	<i>с.м.</i> com.digt.trusted.jce.provider.test.CryptoProCSPKeyStoreTest

## TSPHelper

<b>Соответствует</b>	Получение и проверка штампа времени
<b>Пакет</b>	com.digt.trusted.tsp
<b>Класс</b>	TSPHelper
<b>Совместимость с JCE</b>	Вспомогательный класс для получения/проверки штампа времени с локальной/удаленной службы штампов времени
<b>Описание класса</b>	<ul style="list-style-type: none"> <li>■ public TSPHelper(CMSProcessable msg) – конструктор создает экземпляр</li> </ul>

	<p>объекта, проинициализированный указанными в параметрах данными для последующего получения или проверки штампа времени.</p> <p><i>Параметры:</i> msg - данные, для которых должен быть вычислен data imprint хэш, включаемый в запрос на получение штампа времени.</p> <ul style="list-style-type: none"> <li>■ public byte[] getDataImprint() - возвращает message imprint GOST3411 хэш для данных, переданных через конструктор или метод setDataImprint.</li> <li>■ public void setDataImprint(CMSProcessable msg) - генерирует ГОСТ3411 хэш указанных данных и сохраняет его для дальнейшего использования.</li> </ul> <p><i>Параметры:</i> msg - данные, для которых должен быть вычислен data imprint хэш, включаемый в запрос на получение штампа времени.</p> <ul style="list-style-type: none"> <li>■ public static void validateCert(TimeStampToken tok) - проверяет валидность сертификата, содержащегося в штампе времени, и корректность подписи штампа времени на этом сертификате.</li> </ul> <p><i>Параметры:</i> tok - проверяемый штамп времени</p> <ul style="list-style-type: none"> <li>■ public TimeStampToken getTimeStampLocal(PrivateKey key, X509Certificate ca, X509Certificate cert) - создает штамп времени с помощью встроенного в провайдер генератора. Возвращает штамп времени.</li> </ul> <p><i>Параметры:</i> key - приватный ключ локального TSA ca - сертификат УЦ, на котором был сгенерирован сертификат локального TSA cert - сертификат локального TSA</p> <ul style="list-style-type: none"> <li>■ 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.</li> </ul> <p><i>Параметры:</i> tsaUrl - адрес сервера служб штампов времени.</p> <ul style="list-style-type: none"> <li>■ public static AttributeTable convertToAttr(TimeStampToken tok) - возвращает штамп времени как экземпляр AttributeTable в формате, пригодном для использования в качестве атрибута при генерации подписи.</li> <li>■ 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.</li> </ul> <p><i>Параметры:</i> signer - данные подписчика, в которых предполагается наличие атрибута штампа времени</p>
<b>Методы класса</b>	с.м. com.digt.trusted.tsp.test.TSPTTest

## Техническая поддержка

По вопросам технической поддержки ПО Trusted Java обращайтесь:

По электронной почте: [support@trusted.ru](mailto:support@trusted.ru)

По телефонам: (8362) 55-62-81, (8362) 55-62-27

По адресу: 424019, Россия, Республика Марий Эл, г. Йошкар-Ола, ул. Фестивальная, д.73.

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) 55-62-81

Факс: (8362) 55-62-27

---

Интернет: <http://www.trusted.ru>

E-mail: [info@trusted.ru](mailto:info@trusted.ru)

