127 018, Москва, Сущевский вал, д.18
Телефон: (495) 995 4820
Факс: (495) 995 4820
http://www.CryptoPro.ru
E-mail: info@CryptoPro.ru

Средство Криптографической Защиты Информации

КриптоПро JCP

Версия 2.0

Правила пользования

ЖТЯИ.00091-01 92 01

Листов 33

2016 г.

© ООО "Крипто-Про", 2000-2016. Все права защищены.

Авторские права на средство криптографической защиты информации «КриптоПро JCP» версия 2.0 и эксплуатационную документацию зарегистрированы в Российском агентстве по патентам и товарным знакам (Роспатент).

Документ входит в комплект поставки программного обеспечения СКЗИ «КриптоПро JCP» версия 2.0, на него распространяются все условия лицензионного соглашения. Без специального письменного разрешения ООО "КРИПТО-ПРО" документ или его часть в электронном или печатном виде не могут быть скопированы и переданы третьим лицам с коммерческой целью.

Оглавление

1. Аннотация

Данный документ содержит правила пользования средства криптографической защиты информации (СКЗИ) «КриптоПро JCP» версия 2.0, его состав, назначение, ключевую систему, требования по защите от НСД.

Документ предназначен для администраторов информационной безопасности учреждений, осуществляющих установку, обслуживание контроль за соблюдением требований к эксплуатации средств СКЗИ, а также администраторам серверов, сетевых ресурсов предприятия и других работников службы информационной безопасности, осуществляющим настройку рабочих мест для работы со средствами СКЗИ.

Инструкции администраторам безопасности и пользователям различных автоматизированных систем, использующих СКЗИ «КриптоПро JCP» версия 2.0 должны разрабатываться с учетом требований настоящих Правил.

2. Сведения об изделии

СКЗИ «КриптоПро JCP» версия 2.0 предназначено для защиты открытой информации в информационных системах общего пользования (вычисление и проверка ЭП) и защиты конфиденциальной информации, не содержащей сведений, составляющих государственную тайну, в корпоративных информационных системах (шифрование и расшифрование информации, вычисление и проверка имитовставки, вычисление значения хэш-функции, вычисление и проверка ЭП).

СКЗИ «КриптоПро JCP» версия 2.0 совместимо «КриптоПро CSP» по выполняемым криптографическим функциям и ключам.

При эксплуатации СКЗИ «КриптоПро JCP» версия 2.0 должны выполняться следующие требования:

- Средствами СКЗИ не допускается обрабатывать информацию, содержащую сведения, составляющие государственную тайну. Ключевая информация является конфиденциальной.
- Размещение СКЗИ «КриптоПро JCP» версия 2.0 в помещениях, предназначенных для ведения переговоров, в ходе которых обсуждаются вопросы, содержащие сведения, составляющие государственную тайну, осуществляется установленным порядком.
- Использование СКЗИ для защиты речевой информации без проведения соответствующих дополнительных исследований запрещено. При ведении переговоров, содержащих конфиденциальную информацию, должны отключаться устройства преобразования речи аппаратной компоненты СКЗИ.
- ПЭВМ, на которых используется СКЗИ, должны быть допущены для обработки конфиденциальной информации по действующим в Российской Федерации требованиям по защите информации от утечки по техническим каналам, в том числе, по каналу связи (например, СТР-К), с учетом модели угроз в информационной системе заказчика, которым должно противостоять СКЗИ «КриптоПро JCP» версия 2.0.
- Инсталляция СКЗИ «КриптоПро JCP» версия 2.0 на рабочих местах должна производиться только с дистрибутива, полученного по доверенному каналу.
- Встраивание СКЗИ «КриптоПро JCP» версия 2.0 в другие средства возможно только с использованием функций, указанных в п.10. В случае использования прочих вызовов необходимо производить разработку отдельного СКЗИ в соответствии с действующей нормативной базой (в частности, с Постановлением Правительства Российской Федерации от 16 апреля 2012 г. № 313 и Положением о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)).

СКЗИ «КриптоПро JCP» версия 2.0 при условии выполнения настоящих Правил обеспечивает криптографическую защиту конфиденциальной информации от внешнего нарушителя, самостоятельно осуществляющего создание методов и средств реализации атак, а также самостоятельно реализующего атаки.

1. Порядок распространения СКЗИ

Установочные модули СКЗИ и комплект эксплуатационной документации к нему могут поставляться пользователю Уполномоченной организацией двумя способами:

1. На носителе (CD, DVD - диски);
2. Посредством загрузки через Интернет.

Для получения возможности загрузки установочных модулей СКЗИ и комплекта эксплуатационной документации пользователь направляет свои учетные данные Уполномоченной организации. Учетные данные могут быть направлены посредством заполнения специализированной регистрационной формы на сайте Уполномоченной организации.

После получения Уполномоченной организацией учетных данных пользователю предоставляется доступ на страницу загрузки установочных модулей СКЗИ и комплекта эксплуатационной документации. При загрузке пользователем установочных модулей СКЗИ и комплекта эксплуатационной документации Уполномоченной организацией присваивается учетный номер, идентифицирующий экземпляр СКЗИ, предоставленный пользователю.

Вместе с указанными данными на странице загрузки размещаются контрольные суммы установочных модулей и документации. Контрольные суммы рассчитываются в соответствии с ГОСТ Р 34.11-2012 (256 бит). Пользователь, загрузив установочные модули СКЗИ и эксплуатационную документацию должен убедиться в целостности полученных данных.

Установка СКЗИ на рабочее место пользователя может быть осуществлена только в случае подтверждения целостности полученных установочных модулей СКЗИ и эксплуатационной документации.

Примечание. Средство контроля целостности (cpverify.exe или иное сертифицированное средство) должно быть получено на дистрибутивном носителе доверенным образом или через канал связи, защищенный сертифицированными ФСБ России криптографическими средствами. При этом допускается использование утилиты cpverify.exe из состава СКЗИ «КриптоПро CSP» версии 4.0 при условии, что она была получена на дистрибутивном носителе, а не загружена через общедоступные каналы связи.

---

2. Основные технические данные и характеристики

СКЗИ «КриптоПро JCP» версия 2.0, функционирует под управлением следующих Java-машин:

- Java-машина производства Oracle «Java(TM) 2 Runtime Environment, Standard Edition версии 1.6 и выше на 32-битной и 64-битной платформе.
- Java-машины J9VM производства IBM «Java(TM) 2 Runtime Environment, Standard Edition» версии 1.6 и выше на 32-битной и 64-битной платформе.

Порядок и сроки эксплуатации виртуальных машин, в среде которых функционирует СКЗИ «КриптоПро JCP» версия 2.0, определяются производителями виртуальных машин.

Следующие программные компоненты СКЗИ «КриптоПро JCP» версия 2.0 можно использовать без проведения дополнительных тематических исследований:

- утилита для выработки CMS сообщений cmsutil;
- приложение для создания TLS-туннеля tls_proxy.

Использование СКЗИ «КриптоПро JCP» версия 2.0 с выключенным режимом усиленного контроля использования ключей допускается только в целях тестирования. Включение данного режима описано в п. 11 Руководства администратора безопасности СКЗИ.

4.1 Утилита cmsutil

Утилита cmsutil предназначена для выработки CMS-enveloped сообщений в соответствии с RFC 4490 и Рекомендациями Технического комитета по стандартизации «Криптографическая защита информации» (ТК 26) «Рекомендации по стандартизации. Использование алгоритмов ГОСТ 28147-89, ГОСТ Р 34.11 и ГОСТ Р 34.10 в криптографических сообщениях формата CMS».

Для зашифрования файла с помощью утилиты cmsutil необходимо выполнить следующую команду:

java -jar cmsutil.jar -encrypt -certstore <Название хранилища доверенных сертификатов> [-pass <Пароль от хранилища>] -alias <Имя сертификата> -in <Файл с данными для шифрования> -out <Файл выхода>.

Для расшифрования с помощью утилиты cmsutil необходимо выполнить следующую команду:

java -jar cmsutil.jar -decrypt -keystore <Тип хранилища> -alias <Имя контейнера> [-pass <Пароль контейнера>] -in <Файл с шифрованными данными> -out <Файл выхода>

4.2 Утилита tls_proxy

Утилита tls_proxy представляет собой приложение для создания TLS-туннеля. Данная утилита позволяет устанавливать TLS-соединение в соответствии с Рекомендациями по стандартизации технического комитета по стандартизации «Криптографическая защита информации» (ТК 26) «Рекомендации по стандартизации. Использование наборов алгоритмов шифрования на основе ГОСТ 28147-89 для протокола безопасности транспортного уровня (TLS)».

Утилита может быть запущена следующим образом:

tls_proxy <ListenPort>

При этом, в папке приложения должен лежать конфигурационный файл, устроенный следующим образом:

```
<Config>
<Parameters inactiveTimeout=«60» checkInactiveTimeout=«30» serverSoTimeout=«600» />
<CertStore path=«c:\software\Keys\tomcat7\test_ca.store» password=«1» />
<Addresses>
<Address>
<ListenPort>9000</ListenPort>
<Host>cpca.cryptopro.ru</Host>
<Port>443</Port>
<ClientAuthEnabled>false</ClientAuthEnabled>
</Address>
<Address>
<ListenPort>9001</ListenPort>
<Host>cryptopro.ru</Host>
<Port>4444</Port>
<ClientAuthEnabled>true</ClientAuthEnabled>
<KeyType>HDImageStore</KeyType>
<KeyPassword>Pass1234</KeyPassword>
</Address>
</Addresses>
</Config>
```

Пояснение:

Parameters - технологические параметры соединений:

inactiveTimeout - период времени (сек), после которого подключение считается неактивным и может быть закрыто (время активности обновляется при каждом запросе).

checkInactiveTimeout - период (сек) проверки соединений (для таймера). Неактивные закрываются.

serverSoTimeout - период ожидания (сек) входящих соединений.

CertStore - хранилище доверенных сертификатов. path - путь к хранилищу, password - пароль к нему.

Addresses - блок адресов.

Address - отдельный адрес сервера.

ListenPort - порт, данные из которого транслируются на сервер и обратно. С этим портом запускается приложение tls_proxy. Отдельный экземпляр приложения, слушающего ListenPort, работает с конкретным адресом, связанным с этим портом.

Host, Port - параметры сервера, куда происходит трансляция из ListenPort.

ClientAuthEnabled - флаг, определяющий, включена ли на сервере клиентская аутентификация; если включена, то потребуются параметры KeyType и KeyPassword - тип контейнера и пароль к нему.

1. Ключевая система

СКЗИ «КриптоПро JCP» версия 2.0 является системой с открытым распределением ключей. Ключи проверки электронной подписи и обмена представляются в виде сертификатов открытых ключей.

Сроки действия:

- максимальный срок действия закрытых ключей шифрования и ключей ЭП - 1 год 3 месяца;
- максимальный срок действия открытых ключей шифрования - 1 год 3 месяца;
- срок действия сертификата ключа проверки ЭП пользователя - не больше 5 лет;
- максимальный срок действия ключей проверки ЭП при использовании алгоритма ГОСТ Р 34.10-2001 - 15 лет;
- максимальный срок действия ключей проверки ЭП при использовании алгоритма ГОСТ Р 34.10-2012 - 15 лет.

Ключевой контейнер

При формировании закрытые ключи СКЗИ «КриптоПро JCP» версия 2.0 записываются на ключевой носитель (ключевой контейнер).

Ключевой контейнер может содержать:

- только ключ электронной подписи;
- только ключ шифрования;
- ключ электронной подписи и ключ шифрования одновременно.

Дополнительно ключевой контейнер содержит служебную информацию, необходимую для обеспечения криптографической защиты ключей, их целостности и т. п.

Каждый ключевой контейнер (независимо от типа носителя), является самодостаточным и содержит всю необходимую информацию для работы как с самим контейнером, так и с закрытыми (и соответствующими им открытыми) ключами.

Хранение ключевых носителей

Личные ключевые носители пользователей рекомендуется хранить в сейфе. Пользователь несет персональную ответственность за хранение личных ключевых носителей.

При наличии в организации, эксплуатирующей СКЗИ, администратора безопасности, и централизованном хранении ключевых носителей, администратор безопасности организации несет персональную ответственность за хранение личных ключевых носителей пользователей. Личные ключевые носители администратора безопасности должны храниться в его личном сейфе.

Требования по хранению личных ключевых носителей распространяются на ПЭВМ (в том числе и после удаления ключей с диска).

Настоятельно рекомендуется использовать парольную защиту при хранении ключей на ЖМД.

При необходимости передачи ключевого носителя постороннему, информацию с него необходимо гарантированно удалить.

Уничтожение ключей на ключевых носителях

Ключи на ключевых носителях (включая смарт-карты), срок действия которых истек, уничтожаются путем удаления ключевых контейнеров средствами ПО СКЗИ или с помощью Контрольной Панели, после чего ключевые носители могут использоваться для записи на них новой ключевой информации.

1. Требования по проведению контроля целостности

Контроль целостности JAR-файла провайдера осуществляется при загрузке провайдера посредством проверки подписей трех файлов JCP.jar, ASN1P.jar и asn1rt.jar. Создание и проверка подписей JAR-файлов реализованы в классе JarChecker.

6.1. Создание подписи JAR-файла

Для создания подписи некоторого JAR-файла следует вызвать функцию main класса JarChecker со следующими параметрами:

-sign [-alias keyAlias] [-storetype storeType] [-keypass keyPassword] [-in jar_file] [-out signed_jar_file] [-delsign]

Описание передаваемых параметров:

параметр	значение	обязательный/необязательный
-sign	команда создания новой подписи	обязательный параметр
-alias	имя ключа электронной подписи, на котором осуществляется подпись	обязательный параметр
-storetype	имя ключевого носителя, на котором лежит ключ	по умолчанию - HDImageStore, обязательный если ключ лежит не на жестком диске
-keypass	пароль на ключ подписи	обязательный, если пароль на ключ установлен
-in	полный путь к подписываемому JAR-файлу	обязательный параметр
-out	полный путь к файлу, в который будет записан подписанный JAR-файл	обязательный параметр
-delsign	флаг, определяющий удаление неверных подписей	если указан, то неверные подписи будут удалены

Описание процесса создания подписи:

- в первую очередь осуществляется проверка всех существующих подписей;
- если существуют неверные подписи и указан параметр -delsign, то эти подписи удаляются;
- если после проверки и, возможно, удаления неверных подписей, общее количество подписей равно и превышает 16, то новая подпись создаваться не будет;
- если среди оставшихся подписей уже существует подпись на заданном ключе электронной подписи, и она верна, то новая подпись не создается;

- если среди оставшихся подписей уже существует подпись на заданном ключе электронной подписи, и она не верна, то неверная подпись удаляется (в независимости от флага -delsign) и создается новая подпись на этом ключе;
- в противном случае просто формируется новая подпись.

В процессе формирования подписи JAR-файла осуществляется копирование всех классов, входящих в исходный JAR-файл, в новый JAR-файл, определенный путем -out. После чего в директории META-INF нового JAR-файла создаются два файла: Digest.CP и Sign.CP (если подпись производится не в первый раз, т.е. если такие файлы уже существуют в исходном JAR-файле, то файл Digest.CP не изменяется, а к содержимому Sign.CP добавляется информация о новой подписи).

Первый файл представляется собой набор пар: имя класса, входящего в JAR-файл, но не входящего в директорию META-INF, и значение хэша на содержимое этого класса. Второй файл содержит в себе следующую информацию:

- количество подписей (уже существующих + только что созданная);
- набор подписей (уже существующие + новая подпись);
- набор соответствующих подписям сертификатов (уже существующие + новый).

Новая подпись (как и все предыдущие) производится на содержимое файла Digest.CP.

6.1. Проверка подписи JAR-файла

Для проверки подписи некоторого JAR-файла следует вызвать функцию main класса JarChecker со следующими параметрами:

-verify [-in signed_jar_file] [-cert cert_file]

Описание передаваемых параметров:

параметр	значение	обязательный/необязательный
-verify	команда проверки подписи	обязательный параметр
-in	полный путь к подписанному JAR-файлу	обязательный параметр
-cert	полный путь к сертификату, на котором осуществляется проверка подписи	если указан, то осуществляет проверка подписи, соответствующей заданному сертификату. В противном случае осуществляется проверка всех имеющихся подписей JAR-файла

Описание процесса проверки подписи:

- если в JAR-файле, подпись которого проверяется, не существует файлов Digest.CP и Sign.CP, то файл не содержит подписей и никакой проверки не требуется;
- если такие файлы имеются, то в первую очередь осуществляется подсчет хэшей всех классов подписанного JAR-файла, не входящих в директорию META-INF, и проверка соответствия результата значениям хэшей, содержащихся в файле Digest.CP. Если результаты различны, то выдается Exception (считается, что JAR-файл поврежден);
- если хэши сошлись, то осуществляется собственно проверка подписи:
    - если задан конкретный сертификат (командой -cert), то производится поиск соответствующего сертификата в файле Sign.CP и, если такой сертификат найден, осуществляется проверка соответствующей подписи;
    - в противном случае осуществляется проверка всех подписей, содержащихся в файле Sign.CP. При этом после проверки на экран выводится информация о всех неверных подписях.

6.1. Контроль целостности JAR-файла провайдера

При загрузке провайдера «КриптоПро JCP» версия 2.0 осуществляется проверка подписей трех файлов jcp.jar, ASN1P.jar и asn1rt.jar на заданном ГОСТ сертификате (сертификат "прошит" в классе JarChecker), а также выполняется проверка DSA-подписи на SUN'овском сертификате.

Контроль целостности осуществляется при помощи функции check() класса JarChecker, которая запускается в классе Starter загружаемого провайдера, собранного с отключенным флагом DEBUG. Для корректной работы данной функции сертификат прошит в самом классе JarChecker, и указание пути к файлу с сертификатом при осуществлении контроля целостности не требуется.

6.1. Командная строка CPVerify

Командная строка CPVerify введена для более удобного контроля целостности, а также возможности администрирования систем, в которых отсутствует графическое расширение или пакеты Java AWT и Swing.

С помощью командной строки можно создавать хранилища хэшей, добавлять в них файлы, удалять файлы из хранилища, пересчитывать и проверять хэши файлов в хранилище, а также работать с основным системным хранилищем.

Общий синтаксис вызова

Командной строке Prompt всегда передается следующий набор параметров:

doing repository [params]

Параметр doing определяет действие, требуемое от командной строки. Он обязательно должен стоять первым. Значения параметра приведены в таблице:

параметр	действие
-verify	Проверить файлы в хранилище. Команда проверяет хэши одного или нескольких файлов из указанного хранилища.
-make	Пересчитать хэши файлов в хранилище. По команде пересчитывается хэш одного или нескольких файлов в хранилище.
-add	Добавить файлы в хранилище. Команда добавляет один или несколько файлов в хранилище, и пересчитывает для них хэш.
-delete	Удалить файлы из хранилища. Команда удаляет один или несколько файлов из хранилища.
-create	Создать хранилище. Команда создает новое хранилище. Если хранилище уже существует, то оно будет перезаписано.
-check	Проверить статус хранилища. Команда проверяет статус хранилища: не повреждено или не удалено ли оно.
-setdefault	Сделать хранилище основным системным. Основное системное хранилище - то, в котором хранятся хэши наиболее важных системных файлов, и которое периодически проверяют внутренние службы криптопровайдера.
-getdefault	Узнать, какое хранилище является основным системным.
-print	Вывести состояние всех файлов в хранилище. Команда выводит состояние всех файлов в хранилище: не повреждены ли они, не удалены ли.
-help	Вывести справку. Общая справка по всем командам.

Параметр repository является необходимым во всех командах, кроме -help и -getdefault. Он задает хранилище, с которым будет проводиться операция. Он может быть определен одним из трех следующих способов:

параметр	хранилище
-repfile filename	Хранилище - файл, с именем filename
-reppref	Хранилище расположено в каталоге системных настроек (реестр для Windows).
-repdefault	Основное системное хранилище

Параметр repository может быть любым по счету, но не первым. Если он равен -repfile, следующее слово в командной строке считается именем файла.

Параметры [params] зависят от команды. Для всех команд действует параметр -help - подробная справка по команде. Команды -help, -print, -getdefault, -setdefault, -check, -create не

предполагают дополнительных параметров. В командах -verify, -make, -add, -delete надо определять список файлов, над которыми производится действие, специальными параметрами, указанными ниже в таблице:

Команда	Параметр, определяющий файлы	Действие
-verify	-all	Проверить все файлы, лежащие в хранилище.
	-file filename1 [-file filename2 [...]]	Проверить файлы filename1, filename2,.,если они есть в данном хранилище.
-make	-all	Пересчитать хэши всех файлов, лежащих в хранилище
	-file filename1 [-file filename2 [...]]	Пересчитать хэши файлов filename1, filename2,..., лежащих в хранилище
-add	-file filename1 [-file filename2 [...]]	Добавить в хранилище файлы filename1, filename2,…
-delete	-all	Удалить все файлы из хранилища
	-file filename1 [-file filename2 [...]]	Удалить файлы filename1, filename2,... из хранилища.

Любая команда из тех, которые изменяют хранилище, перед сохранением хранилища вызывает проверку всех файлов, в нем содержащихся. Поэтому если какие-то файлы в хранилище повреждены, его состояние можно перезаписать только удалив их всех из него одной командой -delete, или пересчитав для всех хэш.

Приложение завершается с ошибкой (выход по исключению), если возникла непредвиденная ситуация (отказано в доступе к файлу, ошибка ввода-вывода и т.д.), или если возникла ошибка при работе с хранилищем, когда оно считается существующим. Так, если проверяется статус поврежденного хранилища, приложение завершится нормально, выдав диагностическое сообщение, однако если проводится проверка файлов в поврежденном хранилище, приложение завершится с ошибкой. Приложение завершается нормально, если проверяемый файл из хранилища поврежден, выдавая соответствующую диагностику.

6.2. Список файлов, добавляемых в хранилище для контроля целостности.

Перед началом работы с провайдером необходимо создать основное хранилище в системных настройках, права на изменения которого даны только администратору. В него следует добавить исполняемые файлы Java, файлы содержащие пакеты java.lang, java.security, java.io, javax, системные библиотеки, конфигурационные файлы Java-машины, модули JCP, которые находятся в каталоге [путь к JRE]/lib/ext.

Под контролем целостности должны находится следующие .jar файлы:

AdES-core.jar	CAdES.jar	J6CF.jar
ASN1P.jar	cpSSL.jar	J6Oscar.jar
asn1rt.jar	forms_rt.jar	JCPControlPane.jar

JCPinstGUI.jar	JCPRevTools.jar	Oscar.jar
JCPinst.jar	JCPxml.jar	samples.jar
JCP.jar	JCryptoP.jar	samples-sources.jar
JCPRequest.jar	JCSP.jar	XAdES.jar
JCPRevCheck.jar	OCF.jar	XMLDSigRI.jar

Дополнительно под котнролем целостности должны находится следующие файлы:

Для ОС Windows:

SHELL32.DLL	WS2_32.DLL	WINSCARD.DLL
COMCTL32.DLL	GDI32.DLL	COMDLG32.DLL
ADVAPI32.DLL	KERNEL32.DLL	URLMON.DLL
IMM32.DLL	OLEAUT32.DLL	SECUR32.DLL
WINMM.DLL	WININET.DLL	OLE32.DLL
USER32.DLL	IMAGEHLP.DLL	DSOUND.DLL
PSAPI.DLL	CRYPT32.DLL	MSWSOCK.DLL
VERSION.DLL	WSOCK32.DLL	

Для Linux систем:

libc.so.6	libX11.so.6	libXtst.so.6
libstdc++.so.6	libXext.so.6	libasound.so.2
libdl.so.2	libXft.so.2	libgcc_s.so.1
libm.so.6	libXi.so.6	
libpthread.so.0	libXRender.so.1	

Этот список следует проверять средствами командной строки в отдельном процессе до загрузки рабочей Java-машины не реже одного раза в сутки.

В случае, если CPVerify зафиксирует изменение хотя бы одного элемента Java-машины, использование криптопровайдера следует прекратить до выяснения и устранения причин возникновения ошибки.

1. Требования по защите от НСД

Защита аппаратного и программного обеспечения от НСД при установке и использовании СКЗИ является составной частью общей задачи обеспечения безопасности информации в системе, в состав которой входит СКЗИ.

Наряду с применением средств защиты от НСД необходимо выполнение приведенных ниже организационно-технических и административных мер по обеспечению правильного функционирования средств обработки и передачи информации, а также установление соответствующих правил для обслуживающего персонала, допущенного к работе с конфиденциальной информацией.

Защита информации от НСД должна обеспечиваться на всех технологических этапах обработки информации и во всех режимах функционирования, в том числе при проведении ремонтных и регламентных работ.

Защита информации от НСД должна предусматривать контроль эффективности средств защиты от НСД. Этот контроль должен периодически выполняться администратором безопасности на основе требований документации на средства защиты от НСД.

В организации, эксплуатирующей СКЗИ, должен быть назначен администратор безопасности, на которого возлагаются задачи организации работ по использованию СКЗИ, выработки соответствующих инструкций для пользователей, а также контроль за соблюдением требований по безопасности.

Администратор безопасности не должен иметь возможности доступа к конфиденциальной информации пользователей.

Правом доступа к рабочим местам с установленными СКЗИ должны обладать только определенные для эксплуатации лица, прошедшие соответствующую подготовку. Администратор безопасности должен ознакомить каждого пользователя, применяющего СКЗИ, с документацией на СКЗИ, а также с другими нормативными документами, созданными на её основе.

При использовании СКЗИ «КриптоПро JCP» версия 2.0 также следует принять следующие организационные меры:

1. Провайдер «КриптоПро JCP» версия 2.0 должен использоваться в среде, защищенной от действий внешнего нарушителя, и в корпоративных сетях, защищенных от внутреннего нарушителя.
2. Необходимо ограничить возможность вывода информации с используемой ПЭВМ через порты COM, LPT, USB, IEEE 1394, а также средствами Bluetooth, Wi-Fi и аналогичных.
3. Право доступа к рабочим местам с установленным ПО СКЗИ «КриптоПро JCP» версия 2.0 предоставляется только лицам, ознакомленным с правилами пользования и изучившим эксплуатационную документацию на программное обеспечение, имеющее в своем составе СКЗИ «КриптоПро JCP» версия 2.0.
4. Запретить осуществление несанкционированного администратором безопасности копирование ключевых носителей.
5. Запретить разглашение содержимого ключевых носителей и передачу самих носителей лицам, к ним не допущенным, а также выводить ключевую информацию на дисплей и принтер.

6. Запретить использование ключевых носителей в режимах, не предусмотренных правилами пользования СКЗИ «КриптоПро JCP» версия 2.0, либо использовать ключевые носители на посторонних ПЭВМ.
7. Запретить запись на ключевые носители посторонней информации.
8. Требования по хранению личных ключевых носителей распространяются на ПЭВМ (в том числе и после удаления ключей с диска).
9. При использовании СКЗИ «КриптоПро JCP» версия 2.0 на ПЭВМ, подключенных к общедоступным сетям связи, должны быть предприняты дополнительные меры, исключающие возможность несанкционированного доступа к системным ресурсам используемых операционных систем, к программному обеспечению, в окружении которого функционируют СКЗИ, и к компонентам СКЗИ со стороны указанных сетей.
10. При загрузке ОС должен быть реализован контроль целостности программного обеспечения, входящего в состав СКЗИ «КриптоПро JCP» версия 2.0, самой ОС и всех исполняемых файлов, функционирующих совместно с СКЗИ.
11. Должно быть реализовано физическое затирание содержимого удаляемых файлов, в том числе SWAP файла.
12. Должно быть обеспечено тестирование аппаратуры в объеме самотестирования при перезагрузке ежесуточно.
13. Переставлять реализацию класса Preferences с помощью property java.util.prefs.PreferencesFactory запрещается.
14. Необходимо обеспечить административными мерами контроль доступа к системным и пользовательским настройкам Java-машины.
15. При использовании «КриптоПро JCP» версия 2.0 на платформе Windows Java-машина системные и пользовательские настройки хранит в реестре в разделах HKEY_CURRENT_USER\Software\JavaSoft\Prefs и HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Prefs соответсвенно.
16. На платформах Solaris и Linux Java-машина системные и пользовательские настройки хранит в файловой системе. Системные настройки находятся в каталоге .systemPrefs, положение которого определяется переменной java.util.prefs.systemRoot (по умолчанию /etc/.java). Если он недоступен то .systemPrefs находится в каталоге определенном переменной java.home
17. Пользовательские настройки находятся в каталоге .java/.userPrefs, положение которого определяется переменной java.util.prefs.userRoot Если переменная не задана то каталог .java/.userPrefs находится в каталоге определенном переменной user.home.

1. Нештатные ситуации при эксплуатации СКЗИ

Ниже приведен основной перечень нештатных ситуаций и соответствующие действия персонала при их возникновении.

Таблица 1 - Действия персонала в нештатных ситуациях

№п/п	Нештатная ситуация	Действия персонала
1.	Эвакуация, угроза нападения, взрыва и т.п., стихийные бедствия, аварии общего характера в Центре управления ключевой системой.	Остановить все ЭВМ. Персонал, имеющий доступ к ключам, обязан сдать все имеющиеся у него в наличии ключевые носители администратору безопасности. Администратор безопасности упаковывает все ключевые носители, регистрационные карточки сертификатов открытых ключей пользователей, сертификаты ключей проверки ЭП пользователей в опечатываемый контейнер, который выносит в безопасное помещение или здание. Опечатанный контейнер должен находиться под охраной до окончания действия нештатной ситуации и восстановления нормальной работы аппаратных и программных средств СКЗИ. Администратор безопасности оповещает по телефонным каналам общего пользования всех пользователей о приостановке работы системы. В случае наступления события, повлекшего за собой долговременный выход из строя аппаратных средств СКЗИ, администратор безопасности уничтожает всю ключевую информацию с носителей, находящихся в контейнере.
2.	Компрометация одного из личных ключевых носителей.	Порядок действий при компрометации ключей описан в разделе 9.5 "Действия при компрометации ключей".
3.	Выход из строя первого личного ключевого носителя.	Необходимо сообщить по телефону в УЦ о факте выхода из строя личного ключевого носителя и обеспечить его доставку в УЦ для выяснения причин выхода из строя. Для работы используется второй личный ключевой носитель.
4.	Выход из строя второго личного ключевого носителя (при условии, что первый тоже вышел из строя).	Пользователь, у которого вышли из строя оба личных ключевых носителя, является в УЦ для повторной регистрации (без изменения данных регистрации).
5.	Отказы и сбои в работе аппаратной части АРМ со встроенной СКЗИ.	При отказах и сбоях в работе аппаратной части АРМ со встроенной СКЗИ необходимо остановить работу, по возможности локализовать неисправность и в дальнейшем произвести ремонт в установленном порядке и, при необходимости, переустановку СКЗИ.
6.	Отказы и сбои в работе средств защиты от НСД.	При отказах и сбоях в работе средств защиты от НСД, администратор безопасности, должен восстановить работоспособность средств НСД. При необходимости переустановить программно-аппаратные средства НСД.
7.	Утеря личного ключевого носителя.	Утеря личного ключевого носителя приводит к компрометации хранящегося в нем ключа. Порядок действий при компрометации ключей описан в разделе 9.5

№п/п	Нештатная ситуация	Действия персонала
		"Действия при компрометации ключей".
8.	Отказы и сбои в работе программных средств вследствие не выявленных ранее ошибок в программном обеспечении.	При отказах и сбоях в работе программных средств, вследствие не выявленных ранее ошибок в программном обеспечении, необходимо остановить работу, локализовать по возможности причину отказов и сбоев и вызвать разработчика данного ПО или его представителя для устранения причин, вызывающих отказы и сбои. Перед продолжением работы следует в обязательном порядке осуществить перезапуск компьютера.
9.	Отказы в работе программных средств вследствие случайного или умышленного их повреждения.	При отказах в работе программных средств, вследствие случайного или умышленного их повреждения, лицо, ответственное за безопасность функционирования программных и аппаратных средств, обязано произвести служебное расследование по данному факту с целью установления причины отказа и восстановления правильной работы программных средств в установленном порядке.
10.	Отказы в работе программных средств вследствие ошибок оператора.	При отказах в работе программных средств, вследствие ошибок оператора, оператор сообщает о данном факте лицу, ответственному за безопасность функционирования программных и аппаратных средств. Ответственный за безопасность функционирования программных и аппаратных средств дает соответствующие указания обслуживающему персоналу по восстановлению правильной работы программных средств в установленном порядке.

2. Встраивание СКЗИ

При встраивании СКЗИ «КриптоПро JCP» версия 2.0 в прикладные системы необходимо проводить оценку влияния аппаратных, программно-аппаратных и программных средств сети (системы) конфиденциальной связи, совместно с которыми предполагается штатное функционирование СКЗИ, на выполнение предъявленных к СКЗИ требований в следующих случаях:

1) если информация конфиденциального характера подлежит защите в соответствии с законодательством Российской Федерации;
2) при организации криптографической защиты информации конфиденциального характера в федеральных органах исполнительной власти, органах исполнительной власти субъектов Российской Федерации (далее - государственные органы);
3) при организации криптографической защиты информации конфиденциального характера в организациях независимо от их организационно-правовой формы и формы собственности при выполнении ими заказов на поставку товаров, выполнение работ или оказание услуг для государственных нужд (далее - организации, выполняющие государственные заказы);
4) если обязательность защиты информации конфиденциального характера возлагается законодательством Российской Федерации на лиц, имеющих доступ к этой информации или наделенных полномочиями по распоряжению сведениями, содержащимися в данной информации;
5) при обрабатывании информации конфиденциального характера, обладателем которой являются государственные органы или организации, выполняющие государственные заказы, в случае принятия ими мер по охране ее конфиденциальности путем использования средств криптографической защиты;
6) при обрабатывании информации конфиденциального характера в государственных органах и в организациях, выполняющих государственные заказы, обладатель которой принимает меры к охране ее конфиденциальности путем установления необходимости криптографической защиты данной информации.

Указанную проверку необходимо проводить по ТЗ, согласованному с ФСБ России.

1. Использование программных интерфейсов

Разработка программного обеспечения на основе СКЗИ «КриптоПро JCP» версия 2.0 может производиться без создания новых СКЗИ в случае использования вызовов из перечня ниже в соответствии с документацией.

В случае использования прочих вызовов необходимо производить разработку отдельного СКЗИ в соответствии с действующей нормативной базой (в частности, с Постановлением Правительства Российской Федерации от 16 апреля 2012 г. №313 и Положением о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)).

Перечень вызовов, использование которых при разработке систем на основе СКЗИ «КриптоПро JCP» версия 2.0 возможно без дополнительных тематических исследований:

Метод	**Описание**	**Комментарии**
getInstance() класса KeyPairGenerator	Метод *getInstance()* создает объект генерации ключевой пары ЭП и VKO (генератор)	
getInstance() класса KeyFactory	Метод *getInstance()* создаёт объект, создающий объекты секретных и открытых ключей ЭП и VKO на основе ключевых спецификаций, представляющих собой некоторое внутреннее представление ключевой информации	
getInstance() класса KeyStore	Метод создает объект, осуществляющий доступ к ключевому хранилищу.	
getInstance() класса SecureRandom	Метод создает объект класса генератора случайных чисел	
getInstance() класса Signature	Метод создает объект формирования ЭП в соответствии с указанным идентификатором алгоритма подписи.	
getInstance() класса MessageDigest	Метод создает объект функции хэширования в соответствии с указанным идентификатором алгоритма хэширования	
getInstance() класса KeyAgreement	Метод создает объект, вырабатывающий ключи согласования	
getInstance() класса Mac	Метод создает объект, осуществляющий имитозащиту данных по алгоритму, соответствующему указанному идентификатору.	Разрешена при указании параметра «HMAC_GOSTR3411», «HMAC_GOSTR3411_2012_256» или «HMAC_GOSTR3411_2012_512»
Initialize() класса KeyPairGenerator	Метод *initialize()* осуществляет операцию изменения набора параметров вырабатываемых ключевых пар ЭП или VKO (алгоритм ЭП/VKO, параметры	

Метод	Описание	Комментарии
	используемой группы точек эллиптической кривой, алгоритм хэширования, узла замены для алгоритма хэширования).	
generateKeyPair() класса KeyPairGenerator	Метод *generateKeyPair()* класса KeyPairGenerator осуществляет генерацию ключевой пары	
generatePublic() класса KeyFactory	Метод создает объект открытого ключа на основе спецификации открытого ключа (объекта класса PublicKeySpec), содержащей ключевой материал, или закодированного ключа (в виде объекта класса X509EncodedKeySpec).	
generatePrivate() класса KeyFactory	Метод создаёт объект секретного ключа н основе спецификации секретного ключа (объекта класса PrivateKeySpec), содержащей ключевой материал	
Load() класса KeyStore	Метод осуществляет загрузку содержимого стандартного хранилища JCA в память.	
setKeyEntry() класса KeyStore	Метод осуществляет запись секретного ключа на носитель	
store() класса KeyStore	Метод осуществляет перезапись стандартного хранилища JCA	
getKey() класса KeyStore	Метод осуществляет чтение ключа ЭП с носителя	
setCertificateEntry() класса KeyStore	Метод осуществляет запись сертификата ключа проверки ЭП на носитель Запись сертификата в хранилище	
getCertificate() класса KeyStore	Метод осуществляет чтение сертификата ключа проверки ЭП с носителя Чтение сертификата из хранилища	
*deleteEntry()*класса KeyStore	Метод производит удаление ключевого контейнера с носителя	
reset() класса MessageDigest	Метод переинициализирует объект хэширования для повторного использования после получения хэш-значения предыдущего сообщения. Метод может также изменять параметры хэширования (OID) (при использовании алгоритма хэширования ГОСТ Р 34.11-94)	
Clone() класса MessageDigest	Метод создает точную копию существующего объекта хэширования	
Update() класса MessageDigest	Метод осуществляет обработку хэшируемых данных	
read() класса DigestInputStream	Метод осуществляет обработку хэшируемых данных, получаемых из потока	

Метод	Описание	Комментарии
digest() класса MessageDigest	Метод завершает операцию хэширования и получает хэш-значение сообщения	
valid() класса PrivateKeyUsageExtension	Метод, проверяющий срок действия ключа электронной подписи	
initSign() класса Signature	Метод устанавливает секретный ключ подписания и параметры ЭП	
initVerify() класса Signature	Метод устанавливает открытый ключ проверки электронной подписи и параметры ЭП	Метод разрешается использовать только для объектов открытых ключей, полученных с помощью механизмов PKIX.
setParameter() класса Signature	Метод устанавливает используемую хэш-функцию и её параметры в соответствии с переданным идентификатором.	
update() класса Signature	Метод осуществляет обработку переданных данных хэш-функцией.	
sign() класса Signature	Метод производит завершающее преобразование хэш-функции и производит подпись всего накопленного сообщения.	
verify() класса Signature	Метод производит завершающее преобразование хэш-функции и осуществляет проверку подписи	
generateCertificate() класса CertificateFactory	Метод производит генерацию X.509 сертификатов	
getEncoded() класса Certificate, наследуется X509Certificate	Метод осуществляет кодирование существующего сертификата	
getPublicKey() класса Certificate, наследуется X509Certificate	Метод возвращает ключ проверки ЭП из сертификата	
checkValidity() класса X509Certificate	Метод осуществляет проверку срока действия сертификата X509	
getVersion() класса X509Certificate	Метод возвращает номер версии сертификата X509	
*getSerialNumber()*класса X509Certificate	Метод возвращает серийный номер сертификата X509	
getIssuerDN() класса X509Certificate	Метод возвращает DN эмитента сертификата	
getIssuerX500Principal() класса X509Certificate	Метод возвращает DN эмитента сертификата в формате X500	
getSubjectDN() класса X509Certificate	Метод возвращает DN владельца сертификата	

Метод	Описание	Комментарии
getSubjectX500Principal( ) класса X509Certificate	Метод возвращает DN владельца сертификата в формате X500	
*getNotBefore()*класса X509Certificate	Метод возвращает дату начала действия сертификата	
getNotAfter() класса X509Certificate	Метод возвращает дату окончания действия сертификата	
*getTBSCertificate()*класса X509Certificate	Метод осуществляет DER-кодирование сертификата	
getSignature() класса X509Certificate	Метод возвращает подпись под сертификатом	
getSigAlgName() класса X509Certificate	Метод возвращает название алгоритма подписи под сертификатом	
getSigAlgOID() класса X509Certificate	Метод возвращает OID алгоритма подписи под сертификатом	
getSigAlgParams() класса X509Certificate	Метод возвращает параметры алгоритма подписи под сертификатом в DER-кодировке	
getIssuerUniqueID() класса X509Certificate	Метод возвращает уникальный ID эмитента сертификата	
getSubjectUniqueID() класса X509Certificate	Метод возвращает уникальный ID владельца сертификата	
getKeyUsage() класса X509Certificate	Метод возвращает набор разрешенных областей использования ключа	
getExtendedKeyUsage() класса X509Certificate	Метод возвращает области расширенного использования ключа	
getBasicConstraints() класса X509Certificate	Метод возвращает длину расширения BasicConstraints	
getIssuerAlternativeNames() класса X509Certificate	Метод возвращает список альтернативных имён эмитента сертификата	
getSubjectAlternativeNames() класса X509Certificate	Метод возвращает список альтернативных имён владельца сертификата	
getOID() интерфейса ParamsInterface	Метод возвращает объектный идентификатор параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
getDefault() интерфейса ParamsInterface	Метод возвращает значение объектного идентификатора, установленного в контрольной панели, параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
setDefaultAvailable() интерфейса ParamsInterface	Метод осуществляет проверку наличия необходимых прав для установки новых параметров по умолчанию в контрольную панель.	

Метод	Описание	Комментарии
setDefault(OID def) интерфейса ParamsInterface	Метод устанавливает объектный идентификатор по умолчанию для параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
getOIDs() интерфейса ParamsInterface	Метод получает список допустимых объектный идентификаторов параметров для алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
getDefaultSignParams() класса AlgIdSpec	Метод возвращает параметры алгоритма подписи по умолчанию	
getDefaultDigestParams() класса AlgIdSpec	Метод возвращает параметры алгоритма хэширования по умолчанию	
getDefaultCryptParams класса AlgIdSpec	Метод возвращает параметры алгоритма шифрования по умолчанию	
setKeyUsage() класса GostCertificateRequest	Метод устанавливает значение поля keyUsage в запросе на сертификат, описывающее разрешенные области использования ключа.	
addExtKeyUsage() класса GostCertificateRequest	Метод устанавливает значение поля extKeyUsage в запросе на сертификат, описывающее дополнительные области использования ключа.	
addExtension()класса *GostCertificateRequest*	Метод устанавливает дополнительное расширение в список расширений.	
*setPublicKeyInfo()*класса GostCertificateRequest	Метод осуществляет кодирование и запись в структуру запроса параметров и значения передаваемого открытого ключа субъекта	
*setSubjectInfo()*класса GostCertificateRequest	Метод определяет имя субъекта в формате X.500, устанавливает новое имя субъекта.	
*encodeAndSign()*класса GostCertificateRequest	Метод выполняет кодирование запроса в DER-кодировку и подпись сертификата.	
getEncodedCert() класса GostCertificateRequest	Метод отправляет запрос центру сертификации и получает соответствующий запросу сертификат	
getEncodedCertFromDER() класса GostCertificateRequest	Метод отправляет запрос в DER-кодировке центру сертификации и получает соответствующий запросу сертификата	
printToDER() класса GostCertificateRequest	Метод осуществляет печать подписанного запроса в DER-кодировке в передаваемый PrintStream	
getEncodedCertFromBASE64() класса GostCertificateRequest	Метод осуществляет отправку запроса в кодировке BASE64 центру сертификации и получение соответствующего запросу сертификата	
printToBASE64() класса GostCertificateRequest	Метод осуществляет печать подписанного запроса в BASE64-кодировке в передаваемый PrintStream	
getEncodedRootCert() класса GostCertificateRequest	Метод запрашивает и получает корневой сертификат центра сертификации	

Метод	Описание	Комментарии
getRootCertList() класса CA15GostCertificateRequest	Метод получает список корневых сертификатов УЦ (CA15)	
sendCertificateRequest () класса CA15GostCertificateRequest	Метод отправляет в УЦ запрос на сертификат.	
getCertificateRequestList () класса CA15GostCertificateRequest	Метод получает список запросов на сертификат и статуса их обработки.	
checkCertificateStatus () класса CA15GostCertificateRequest	Метод получает статус обработки УЦ ранее отправленного запроса на сертификат.	
getCertificateRequestId() класса CA15GostCertificateRequest	Метод получает строки-идентификаторы запроса на сертификат.	
*getCertificateByRequestId()*класса CA15GostCertificateRequest	Метод получает выпущенный сертификат по идентификатору запроса.	
init() класса Mac	Метод, устанавливающий алгоритм шифрования	
clone() класса Mac	Метод, производящий копирование объекта имитозащиты	
update() класса Mac	Метод, осуществляющий обработку данных в процессе вычисления имитовставки	
doFinal() класса Mac	Метод, завершающий вычисление значения имитовставки	
reset() класса *Mac*	Метод, осуществляющий переинициализацию объекта класса Mac после окончания вычисления значения имитовставки предыдущего сообщения	
verify() класса CAdESSignature	Метод осуществляет проверку всех подписей CAdES	
getCAdESSignerInfos() класса CAdESSignature	Метод получает список всех подписантов	
setCertificateStore() класса CAdESSignature	Метод устанавливает набор сертификатов, которые будут упакованы в подписанное сообщение	

Метод	Описание	Комментарии
setCRLStore() класса CAdESSignature	Метод устанавливает набор списков отозванных сертификатов, которые будут упакованы в подписанное сообщение	
*addSigner()*класса CAdESSignature	Метод осуществляет добавление подписи в формируемое подписанное сообщение.	
open() класса CAdESSignature	Метод осуществляет открытие потока подписываемых данных	
update() класса CAdESSignature	Метод осуществляет обработку порции данных	
close() класса CAdESSignature	Метод осуществляет завершение обработки подписываемых данных.	
replaceSigners() класса CAdESSignature	Метод, осуществляющий изменения данных подписывающих в упакованном подписанном сообщении.	
addRecipient() класса EnvelopedSignature	Метод, добавляющий информацию о получателе сообщения, и подготавливающий сообщение к зашифрованию.	
open() класса EnvelopedSignature	Метод, осуществляющий открытие потока зашифрования сообщения	
update() класса EnvelopedSignature	Метод, осуществляющий шифрование порции данных.	
close() класса EnvelopedSignature	Метод, осуществляющий шифрование финальной порции данных и закрывающий поток.	
getRecipients() класса EnvelopedSignature	Метод, возвращающий список получателей сообщения	
decrypt() класса EnvelopedSignature	Метод, позволяющий получателю расшифровать сообщение	
getCAdESSignatureType() класса CAdESType	Метод возвращает тип CAdES сообщения	
getSignerInfo() класса CAdESSigner	Метод возвращает информацию о подписанте	
getSignerSignedAttributes() класса CAdESSigner	Метод возвращает список подписываемых атрибутов сообщения	
getSignerUnsignedAttributes() класса CAdESSigner	Метод возвращает список неподписываемых атрибутов сообщения	
getSignatureTimestampToken() класса CAdESSigner	Метод возвращает внутренний штамп (signature-timestamp) времени сообщения	
getCAdESCTimestampToken() класса CAdESSigner	Метод возвращает внешний штамп (CAdES-C-timestamp) времени сообщения	

Метод	Описание	Комментарии
getCAdESCertificates() класса CAdESSigner	Метод возвращает список сертификатов (certificate-values)	
getSignerCertificate() класса CAdESSigner	Метод возвращает сертификат подписавшего сообщение	
getSignatureType() класса CAdESSigner	Метод возвращает тип подписи	
getCAdESCountersignerInfos() класса CAdESSigner	Метод возвращает список заверителей сообщения	
setCertificateStore() класса CAdESSigner	Метод устанавливает хранилище сертификатов, из которого позже может быть достан сертификат подписи	
enhance() класса CAdESSigner	Метод осуществляет «усовершенствование» подписи CAdES-BES до CAdES-X Long Type 1	
addCountersigner() класса CAdESSigner	Метод, осуществляющий добавление заверяющей подписи к отдельному подписанту.	
verify() класса CAdESSigner	Метод, осуществляющий проверку одной отдельной подписи CAdES	
build() класса CertPathBuilder	Метод, осуществляющий построение цепочки сертификатов	
getAlgorithm() класса CertPathValidator	Метод, возвращающий имя алгоритма, осуществляющего проверку цепочек сертификатов.	
getDefaultType() класса CertPathValidator	Метод, возвращающий имя алгоритма, осуществляющего проверку цепочек сертификатов, принятого по умолчанию.	
getInstance() класса CertPathValidator	Метод, возвращающий объект класса CertPathValidator.	
getProvider() класса CertPathValidator	Метод, осуществляющий получение объекта провайдера алгоритма, осуществляющего проверку цепочек сертификатов, принятого по умолчанию.	
validate() класса CertPathValidator	Метод, осуществляющий проверку цепочек сертификатов.	
addCertPathChecker() класса CertPathParameters	Метод, устанавливающий дополнительные правила проверки сертификатов.	
*addCertStore()*класса CertPathParameters	Метод, устанавливающий дополнительные хранилища сертификатов и CRL.	
getCertPathCheckers() класса CertPathParameters	Метод, возвращающий список установленных дополнительных правил проверки сертификатов.	

Метод	Описание	Комментарии
getCertStores() класса CertPathParameters	Метод, возвращающий список хранилищ сертификатов и CRL.	
getDate() класса CertPathParameters	Метод, возвращающий дату, на которую проверяется верность цепочки сертификатов.	
getInitialPolicies() класса CertPathParameters	Метод, возвращающий список OID'ов политик, которые должны быть указаны в сертификатах, из которых строится цепочка.	
getPolicyQualifiersRejected() класса CertPathParameters	Метод, возвращающий флаг PolicyQualifiersRejected.	
getSigProvider() класса CertPathParameters	Метод, получающий имя провайдера ЭП.	
getTargetCertConstraints() класса CertPathParameters	Метод, возвращающий ограничения на целевой сертификат.	
getTrustAnchors() класса CertPathParameters	Метод, получающий список доверенных сертификатов.	
isAnyPolicyInhibited() класса CertPathParameters	Метод, возвращающий признак обработки всех политик, указанных в сертификате.	
isExplicitPolicyRequired() класса CertPathParameters	Метод, возвращающий признак наличия явных описаний политик.	
isPolicyMappingInhibited() класса CertPathParameters	Метод, возвращающий признак подавления PolicyMapping.	
isRevocationEnabled() класса CertPathParameters	Метод, устанавливающий признак необходимости проверки сертификатов с помощью OCSP/CRL.	
setAnyPolicyInhibited() класса CertPathParameters	Метод, устанавливающий признак обработки всех политик, указанных в сертификате.	
setCertPathCheckers() класса CertPathParameters	Метод, устанавливающий набор дополнительных правил проверки сертификатов.	
setCertStores() класса CertPathParameters	Метод, устанавливающий набор дополнительных хранилищ сертификатов и CRL.	
setDate() класса CertPathParameters	Метод, устанавливающий дату, на которую должна производиться проверка цепочки сертификатов.	
setExplicitPolicyRequired() класса CertPathParameters	Метод, устанавливающий признак наличия явных описаний политик.	
setInitialPolicies() класса CertPathParameters	Метод, устанавливающий список OID'ов политик, которые должны быть указаны в сертификатах, из	

Метод	Описание	Комментарии
	которых строится цепочка.	
setPolicyMappingInhibited() класса CertPathParameters	Метод, устанавливающий признак подавления PolicyMapping.	
setPolicyQualifiersRejected() класса CertPathParameters	Устанавливает флаг PolicyQualifiersRejected.	
setRevocationEnabled() класса CertPathParameters	Метод, устанавливающий признак необходимости проверки сертификатов с помощью OCSP/CRL.	
setTargetCertConstraints() класса CertPathParameters	Метод, устанавливающий ограничения на целевой сертификат.	
setTrustAnchors() класса CertPathParameters	Метод, устанавливающий список доверенных сертификатов.	
getMaxPathLength() класса PKIXBuilderParameters	Метод, получающий максимальную возможную длину строимой цепочки.	
setMaxPathLength() класса PKIXBuilderParameters	Метод, устанавливающий максимальную возможную длину строимой цепочки.	
getXAdESSignerInfos() класса XAdESSIgnature	Метод получает список всех подписантов	
addSigner() класса XAdESSIgnature	Метод осуществляет добавление подписи в формируемое подписанное XML сообщение.	
open() класса XAdESSIgnature	Метод осуществляет открытие потока подписываемых данных	
update() класса XAdESSIgnature	Метод осуществляет обработку порции данных	
close() класса XAdESSIgnature	Метод осуществляет завершение обработки подписываемых данных.	
verify() класса XAdESSIgnature	Метод осуществляет проверку всех подписей XAdES	
getSignatureType() класса XAdESSigner	Метод, возвращающий типа XAdES-подписи.	
getEarliestValidSignatureTimeStampToken() класса XAdESSignerT	Метод, возвращающий самый ранний валидный внутренний штамп времени.	
verify() класса XAdESSigner	Метод, осуществляющий проверку одной отдельной подписи XAdES	
getSignerInfo()	Метод, возвращающий узел подписи в документе.	

Метод	Описание	Комментарии
класса XAdESSigner		
getSignerCertificate() класса XAdESSigner	Метод, возвращающий сертификат ключа ЭП данного подписанта.	
getElement() класса XAdESSigner	Метод, возвращающий подписываемый узел.	
getDocument() класса XAdESSigner	Метод, возвращающий подписываемый документ.	