ПАК «Звезда»
Техническое описание криптосервиса

Оглавление

СПИСОК СОКРАЩЕНИЙ

АРМ	Автоматизированное рабочее место
ПАК	Программно-Аппаратный Комплекс
СКЗИ	Средство криптографической защиты информации
УВК	Устройство для взаимодействия с криптомодулем
ЭБ	Элемент Безопасности
ЭБКС	Элемент Безопасности криптосервиса

1. ВВЕДЕНИЕ

Настоящий документ описывает устройство, программный интерфейс и сценарии использования криптографического сервиса (криптосервиса) в составе ПАК «Звезда». Архитектура ПАК «Звезда» и используемая терминология описана в документе «Общее описание ПАК «Звезда»» см. [5] .

Криптосервис выполняет функции, связанные с криптографической защитой данных, передаваемых от клиентов сети IOT (далее, *клиентов*) серверу приложений и в обратном направлении. Криптосервис предназначен для использования в составе криптосервера

Далее будем называть криптосервер средой функционирования криптосервиса.

Криптосервис представляет собой программу разработанную для ОС Windows в виде исполняемого файла (EXE)[1].

Криптосервис реализует следующие функции:

- Формирование исходящих сообщений для защищенной передачи прикладных данных клиенту.
- Обработка входящих сообщений от клиента, проверка целостности, расшифровка и извлечение прикладных данных
- Управление криптографическими ключами ***сервера***
- Удаленное управление криптографическими ключами криптомодулей на ***клиентах***.
- Подготовка данных для персонализаци криптомодулей ***клиентов***.
- Генерация ЭП от данных прикладного уровня, приходящих от ***клиента***
-
- Генерация/Выгрузка запросов на сертификат/сертификатов ключей ***клиентов*** и ***сервера***
- Загрузка сертификатов ключей ***клиентов*** и ***сервера*** подписанных внешним УЦ
- Экспорт/импорт различных данных криптосервиса для передачи/копирования на другие устройства с обеспечением конфиденциальности и (опционально) целостности передачи секретных ключей

Криптосервис предоставляет пользовательский интерфейс, разделенный на несколько групп:

Таблица 1. Группы интерфейсов криптосервиса

Название	Описание
RadioApi	Интерфейс для взаимодействия с клиентом. Обеспечивает шифрование/расшифрование CRISP-сообщений
AppApi	Интерфейс для взаимодействия с приложением. Обеспечивает отправку/прием пользовательских данных
AdminApi	Интерфейс для администрирования. Обеспечивает управление ключами и данными клиентов и криптосервиса, персонализация криптомодуля, генерация сертификатов, экспорт/импорт ключей и данных

Каждая группа состоит из 2 интерфейсов – один используется для вызова функций криптосервиса потребителем (суффикс Service), другой – для вызова функций потребителя криптосервисом (суффикс Feedback):

[1] На последующих этапах возможна разработка аналогичного сервиса для ОС семейства UNIX

Порт	Направление	Описание
RadioService	Client -> CS	Прием CRISP-сообщений от клиента
RadioFeedback	Client <- CS	Отправка CRISP-сообщений клиенту
AppService	CS <- App	Отправка данных от приложения клиенту
AppFeedback	CS ->App	Прием данных от клиента, уведомления о доставке
AdminService	Admin ->CS	Функции администрирования криптосервиса
AdminFeedback	Admin <- CS	Уведомления об изменении состояния криптосервиса для мониторинга

Обращения к функциям API осуществляется согласно протоколу RPC, описанному в [8] . Технически каждому интерфейсу соответствует 2 TCP-порта , которые открываются на криптосервисе. По первому производится вызов функций, по второму возвращается ответ. Для использования интерфейса пользователь должен подключиться к соответствующему порту (соответствие порта и интерфейса задается в настройках криптосервиса). Согласно протоколу RPC, криптосервис может обслуживать не более одного соединения для каждого интерфейса.

Далее по документу для указания вызова функции по какому-либо интерфейсу будет использоваться нотация **<имя интерфейса>.<имя функции>**. Например, **RadioService.ProcessUplinkMessage**.

Все важные события, происходящие в криптосервисе, логируются в файл crypto_service.log.

К таким событиям относится:

- Этапы инициализации/завершения работы криптосервиса
- Вход/выход пользователя
- Ошибочные ситуации при удаленном управлении ключами клиентов

Запись лога содержит:

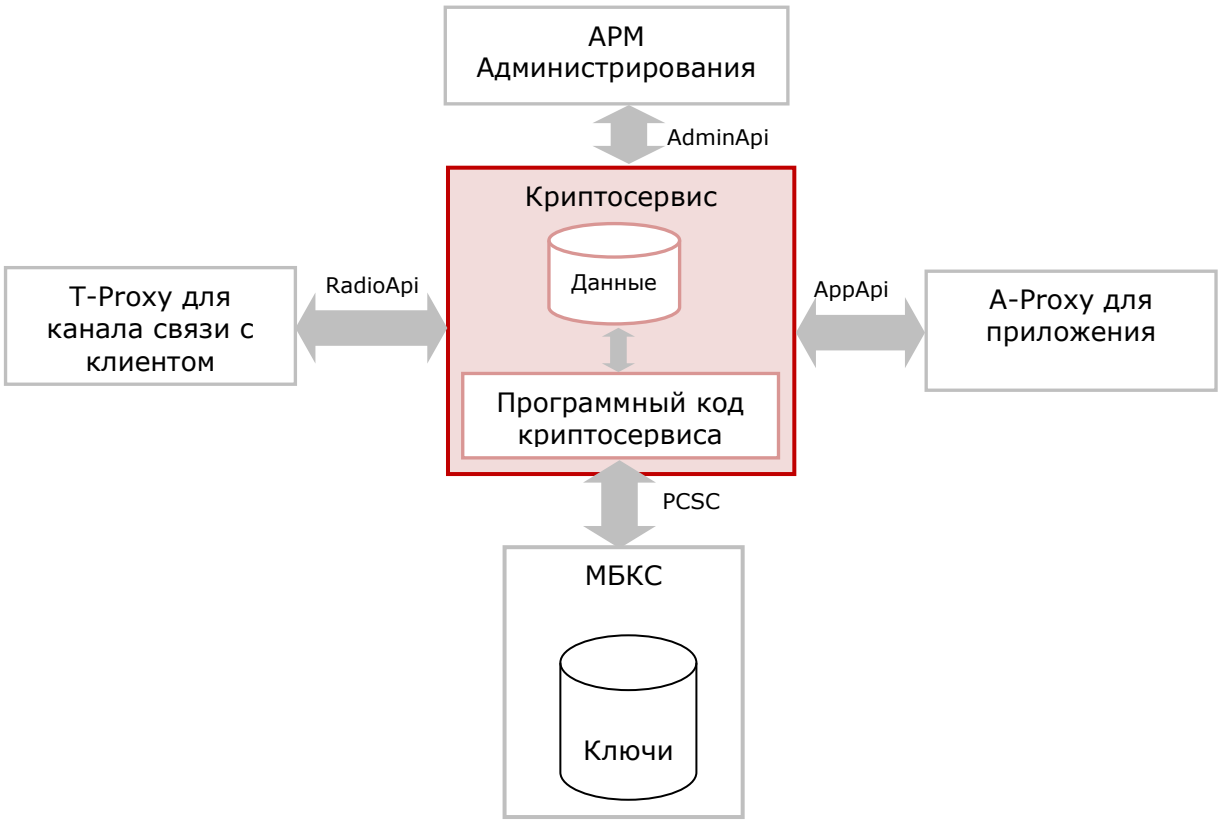
- метку времени,
- информацию о событии

Доступ к некоторым функциям (см. Таблица 5), использующим секретные ключи модуля безопасности, требует особых полномочий. Контроль доступа к таким функциям осуществляется модулем безопасности криптосервера.

Вопросы разграничения доступа к прочим функциям криптосервиса, а также вопросы защиты канала между средой функционирования и криптосервисом выходят за рамки данного документа, и должны при необходимости обеспечиваться средой функционарования криптосервиса.

2. АРХИТЕКТУРА КРИПТОСЕРВИСА

Рисунок 1. Архитектура криптосервиса



Модуль безопасности криптосервера (МБКС) - СКЗИ предназначенное для безопасного хранения ключей и выполнения криптографических операций (подробности в п. **2.1**).

Криптосервис – предоставляет программные интерфейсы для обеспечения защищенного обмена с клиентами, удаленного управления ключами клиентов и администрирования. Для безопасного хранения ключей и выполнения криптографических операций используется МБКС. Криптосервис включает в себя:

- **Программный код криптосервиса** – реализует бизнес-логику криптосервиса. Предоставляет API, посредством которого пользователи криптосервиса взаимодействуют с криптосервисом (см. п.13). Программный код криптосервиса взаимодействует с МБКС для выполнения криптографических операций . В процессе своей работы программный код обращается к БД для чтения/записи необходимой информации и к МБКС для выполнения криптографических операций .
- **Данные** – хранилище информации о каждом подключенном клиенте и служебной информации, относящейся к самому криптосервису.

T-Proxy для канала связи с клиентом – обеспечивает передачу и прием сообщений от подключенных клиентов. Может быть реализована как прокси-сервер или шлюз, обеспечивающий интеграцию криптосервиса с каналом связи с клиентами сети IOT.

A-Proxy для приложения – обеспечивает передачу и прием прикладных данных от подключенных приложений. Может быть реализована как прокси-сервер или шлюз, обеспечивающий интеграцию криптосервиса с серверами приложений или другими потребителями данных от клиентов.

2.1. Модуль безопасности криптосервера

Модуль безопасности криптосервера используется криптосервисом для хранения ключей и выполнения криптографических операций с ключами.

МБКС может быть реализован следующими способами:

Тип МБКС	Применение
плата PCI-Express	Плата PCI-Express предназначена для стационарных серверов с большим количеством клиентов.
чипы в корпусе LGA-40	Для компактных серверных решений в возможностью установки корпуса на плату.
смарт-карты	для компактных серверных решений, снабженных считывателем смарт-карт
сим-карты 2FF (SIM), 3FF (micro-SIM), 4FF (nano-SIM), MFF2 (eSIM)	для компактных серверных решений, снабженных холдером сим-карт

МБКС включает в себя несколько Элементов Безопасности Криптосервера (ЭБКС). Количество ЭБКС может быть различным. Тип МБКС PCI-Express содержит в своем составе несколько ЭБКС.

При использовании типов МБКС с одной микросхемой (таких как LGA-40, смарт-карта, сим-карта), рекомендуется использовать не менее двух микросхем (ЭБКС).

Использование нескольких ЭБКС (кластера) дает следующие преимущества:

- Обеспечение отказоустойчивости на уровне микросхемы. При использовании кластера криптосервис проводит дополнительные операции по резервному копированию определенных ключей модуля безопасности, потеря которых может привести к необходимости переперсонализации всех клиентских устройств. Тем самым, обеспечивается защита от выхода из строя одного или нескольких чипов кластера.
- Повышение производительности за счет параллельного выполнения криптографических операций

Для взаимодействия со всеми типами МБКС используется интерфейс PCSC. Каждое устройство ЭБКС в составе МБКС – отдельное PCSC-устройство.

3. КЛЮЧЕВАЯ СИСТЕМА

Основное назначение криптосервиса - удовлетворение требований целостности, аутентичности, конфиденциальности и неотказуемости при передаче данных между клиентами и криптосервером. Для обеспечения этой функциональности криптосервис (посредством МБКС) хранит и использует следующие криптографические ключи:

Таблица 2. Назначения ключа

Назначение	Описание	Тип ключа (С–секретный ключ, П–ключевая пара)
MK_{BASE}	Мастер-ключ для вывода базового ключа клиента для протокола Crisp	С
MK_{ADM}	Мастер-ключ для вывода ключа администрирования клиента	С
ZMK	Зональный мастер-ключ для экспорта/импорта данных	С
$K_{DS.CS}$	Ключ криптосервиса для ЭП	П
$K_{KM.CS}$	Ключ криптосервиса для управления ключами клиентов	П
$K_{CA.CS}$	Ключ криптосервиса для подписи сертификатов	П
$K_{EXCH.CS}$	Ключ обмена ключами ZMK	П

В криптосервисе информация о всех ключах, вне зависимости от их назначения и прочих свойств, хранится в **ключевых контейнерах[2].**

Для каждого контейнера хранятся следующие параметры:

- Назначение ключа (**`KeyDest`**) – см. Таблица 2.
- Имя контейнера (**`KeyContainerName`**) [опционально]
- Временной лимит использования ключа (**`KeyValidityLen`**) [опционально]
- Список версий ключа
- Дополнительные параметры в зависимости от назначения ключа (см. подразделы данной главы)

Криптосервис может хранить множество ключей различных типов. Максимальное количество ключей ограничивается ресурсами модуля безопасности криптосервера.

Криптосервис предоставляет функции для создания, удаления, и поиска контейнеров по параметрам.

При создании контейнера есть возможность указать его имя. Это может быть использовано для создания различных контейнеров с одним и тем же назначением. Создание двух контейнеров с одинаковым именем и назначением запрещается криптосервисом.

В каждом контейнере может храниться множество ключей. Для каждого ключа хранится следующая информация:

- Уникальный полный идентификатор ключа (**`KeyID`**)
- Номер версии ключа (**`KeyVersion`**)
- Дата выпуска ключа (**`KeyIssuerDate`**)
- Дата истечения срока действия ключа (**`KeyExpirationDate`**)
- Состояние ключа (**`KeyState`**) –
    - INIT – ключевая версия создана, но еще не готова к использованию (например, значение не сгенерировано на клиенте). Операции с ключом и экспорт/импорт недоступны.

[2] Далее по тексту для обозначения ключевого контейнера может быть использовано сокращение «контейнер»

    - ACTIVE – ключ активен. Операции с ключом и экспорт/импорт доступны,
    - REVOKED – ключ отозван (ключ был явно отозван функцией **RevokeKey** или истек срок годности). Операции с ключом и экспорт/импорт запрещены.
- Текстовое описание ключа
- Информация о ключе (см. подразделы данной главы). Содержимое этой информации зависит от назначения и типа ключа. Например, в контейнерах для ключевых пар библиотека может хранить сертификаты

Криптосервис может хранить несколько ключей каждого типа. Ограничения на максимальное количество хранимых ключей для каждого типа ключа определяются конфигурацией используемого ЭБКС. Криптосервис предоставляет функции для генерации новых версий, отзыва и удаления устаревших версий и поиска версий по параметрам.

Для всех ключей криптосервиса характерны следующие особенности:

- Ключевой контейнер создается командой **AdminService.CreateKeyContainer**.
- Ключ генерируется командой **AdminService.GenerateServerKey**.
- Несколько ключей одновременно могут быть в активном состоянии
- Ключ криптосервиса отзывается (переход в состояние REVOKED) командой **AdminService.RevokeServerKey**. Функция генерирует отложенные задания на генерацию/смену ключа для всех клиентов, использующих данный ключ, и ключ будет отозван после сеанса удаленного управления ключами клиента. По результатам выполнения данной задачи криптосервис обновит состояние ключа.
- Ключ удаляется из криптосервиса командой **AdminService.RemoveServerKey**
- Информацию о ключевом контейнере можно получить с помощью функций **AdminService.GetKeyContainerInfo**, **AdminService.GetKeyContainersInfo**
- Информацию о ключах контейнера можно получить с помощью **AdminService.GetKeyContainerKeys**, **AdminService.GetKeyContainerCurrentKey**

При создании контейнера пользователь может задать ограничения на срок годности ключа (**KeyValidityLen**). Если оно задано, то при генерации каждой версии ключа криптосервис автоматически регистрирует дату/время генерации (согласно системному времени) и дату/время окончания срока (исходя из параметра **KeyValidityLen**). При достижении окончания срока версия ключа блокируется (переходит в состояние REVOKED). Если параметр **KeyValidityLen** для контейнера не задан, то версии не имеют срока годности и проверка не производится.

Для удобства работы с ключами каждая версия ключа контейнера имеет собственный идентификатор **KeyID**, который является уникальным в рамках криптосервиса (т.е. идентифицирует как контейнер, так и версию ключа). Идентификатор ключа используется во всех функциях API, в которых нужно указать конкретный ключа. Для получения идентификатора нужно использовать функцию **AdminService.GetKeyContainerKeys** или **AdminService.GetKeyContainerCurrentKey** и выбрать необходимый ключ по его характеристикам, таким как версия, срок действия, описание и т.д..

Криптосервис не накладывает никаких ограничений на имена контейнеров кроме их уникальности для указанного назначения. Однако, рекомендуется именовать контейнеры согласно их дополнительным бизнес-параметрам, выходящим за рамки настоящего описания (если такие имеются)[3].

[3] В настоящее время, при работе с криптосервисом через АРМ администрирования, для каждого типа ключа фактически создается всего один контейнер с пустым именем.

Например:

- KeyDest = K_{BASE}, KeyContainerName = «Segment3» - контейнер ключей K_{BASE} для сегмента № 3
- KeyDest = K_{BASE}, KeyContainerName = «TK»- контейнер временных (транспорнтных) ключей K_{BASE} для первичной персонализации
- KeyDest = K_{BASE}, KeyContainerName = «Company1»- контейнер ключей K_{BASE} для обслуживания клиентов компании «Company1»
- KeyDest = ZMK, KeyContainerName = « SERVER234» - контейнер ключей ZMK для обмена данными с сервером «SERVER234»

3.1. Ключи MK_{BASE}

Контейнер с назначением MK_{BASE} необходим для выработки рабочих ключей взаимодействия с клиентами по протоколу Crisp (K_{BASE}).

Контейнер имеет следующие дополнительные параметры:

- алгоритм диверсификации ключа (**KeyDiversAlg**). Возможные значения:
  - KDF_GOSTR3411 – диверсификация мастер-ключа согласно алгоритму ГОСТ-Р3411.
  - KDF_GOSTR3412 – диверсификация мастер-ключа согласно алгоритму ГОСТ-Р3412.
  - KDF_GOST28147 – диверсификация мастер-ключа согласно алгоритму ГОСТ-28147.
  - KDF_NO_DIVERS – диверсификация мастер-ключа не производится. Ключ используется непосредственно.

Для каждой версии ключа хранится следующая дополнительная информация

- значение секретного ключа (в ЭБКС).

При наличии большого количества зарегистрированных клиентов рекомендуется создать несколько ключевых контейнеров MK_{BASE} для уменьшения нагрузки на каждый из ключей.

При необходимости использования транспортных ключей необходимо создать дополнительный контейнер транспортных ключей MK_{BASE}, которые будут загружаться клиентам при инициализации и заменяться при начале эксплуатации на боевые с помощью функции **AdminService.ChangeClientWorkKey** с указанием “боевого” мастер-ключа. Для транспортных ключей может использоваться параметр **KeyDiversAlg** = KDF_NO_DIVERS

3.2. Ключи MK_{ADM}

Контейнер с назначением MK_{ADM} необходим для выработки ключей администрирования клиента (K_{ADM}).

Контейнер имеет следующие дополнительные параметры:

- алгоритм диверсификации ключа (**KeyDiversAlg**). Возможные значения:
  - KDF_GOSTR3411 – диверсификация мастер-ключа согласно алгоритму ГОСТ-Р3411.
  - KDF_GOSTR3412 – диверсификация мастер-ключа согласно алгоритму ГОСТ-Р3412.
  - KDF_GOST28147 – диверсификация мастер-ключа согласно алгоритму ГОСТ-28147.
  - KDF_NO_DIVERS – диверсификация мастер-ключа не производится. Ключ используется непосредственно.
- количественный лимит пользователей ключа (**KeyUsageCount**).

Для каждой версии ключа хранится следующая дополнительная информация

- значение секретного ключа (в ЭБКС)

При наличии большого количества зарегистрированных клиентов рекомендуется создать несколько ключевых контейнеров $МК_{ADM}$ для уменьшения нагрузки на каждый из ключей.

3.3. Ключи ZMK

Ключи ZMK используются при экспорте/импорте данных.

При необходимости обмена данными с другими участниками системы (см. функции **`AdminService.Export*`** и **`AdminService.Import*`**) необходимо для каждого участника, с которым предстоит обмен, создать ключевой контейнер ZMK, сгенерировать версию ключа (см. 3.8.1.1) и синхронизировать ее с этим участником . Обмен ключами ZMK производится с использованием специального ключа $К_{EXCH.CA}$ (подробности см. раздел 9.1.1).

Пример параметров контейнера ZMK:

- KeyDest = ZMK, KeyContainerName = “Server2”, где Server2 – строковое представление ID другого криптосервиса, с которым вырабатывается общий ключ ZMK.

Для каждой версии ключа хранится следующая дополнительная информация

- значение секретного ключа для шифрования (в ЭБКС).
- значение секретного ключа для расчета криптографической контрольной суммы (в ЭБКС).

3.4. Ключ для формирования ЭП

Контейнер с назначением $К_{DS.CS}$ используется для формирования ЭП сервера от данных, полученных от клиента. Для использования сервиса необходимо создать ключевой контейнер $К_{DS.CS}$ и версию ключа (см. 3.8.1.1).

При генерации ключа ЭП в действительности в МБКС генерируется набор физических ключей (KeySet), размер которого соответствует количеству чипов в составе МБКС. Это позволяет распределять нагрузку на чипы при проведении затратной по времени операции вычисления ЭП.

После генерации кластера ключей можно сгенерировать сертификаты ключей кластера одним из двух способов:

1. Генерация сторонним УЦ - необходимо сгенерировать запросы на сертификаты ключей кластера (**`AdminService.GenCertRequest`**). После генерации сертификатов для каждого ключа кластера сторонним УЦ на основе запросов на сертификат, нужно загрузить список сертификатов вкриптосервис функцией (**`AdminService.LoadCert`**).
2. Генерация встроенным УЦ. (Для данного варианта необходимо предварительно сгенерировать ключ и сертификат $К_{CA.CS}$ (см. 3.6)) - необходимо вызвать функцию GenCert.

Для каждой версии ключа хранится следующая дополнительная информация:

- Закрытые ключи для каждого ключа кластера (в ЭБКС)
- Значение открытого ключа для каждого ключа из набора (после генерации ключа)
- Запрос на сертификат для каждого ключа кластера (после вызова функции **`AdminService.GenCertRequest`**) или сертификат для каждого ключа кластера (после вызова функции **`AdminService.LoadCert`**)

3.5. Ключ для управления ключами клиентов

Контейнер с назначением $K_{KM.CS}$ используется для удаленного управления ключами клиента (см. функции **AdminService.ChangeClientWorkKey**). Для использования функций удаленного управления необходимо предварительно создать ключевой контейнер $K_{KM.CS}$, сгенерировать версию ключа и указать соответствующий **KeyID** при регистрации клиента (параметр **k_cs_id**).

Допускается также и удаленная загрузка ключа $K_{KM.CS}$ на клиента. В этом случае при персонализации клиенту не загружается $K_{KM.CS}$, при регистрации указывается **k_cs_id = 0**, и при эксплуатации на сервере проводится процедура загрузки $K_{KM.CS}$ (вызов функции **AdminService.ChangeServerKeyOnClient** для клиента с указанием старой версии = 0)

При генерации версии ключа $K_{KM.CS}$ в действительности в МБКС генерируется набор физических ключей (KeySet), размер которого соответствует количеству чипов в составе МБКС. Это позволяет

- Обеспечить резервирование ключей $K_{KM.CS}$ для отказоустойчивости системы в случае выхода из строя одного из чипов ЭБКС. Резервирование осуществляется посредством кросс-сертификации всех ключей связки, что позволяет в случае выхода из строя одного из чипов незамедлительно сформировать команды смены ключа $K_{KM.CS}$ на альтернативный ключ того же кластера на всех использующих его клиентах.
- распределять нагрузку на чипы при проведении затратных по времени операций управления ключами.

Для набора ключей хранится следующая дополнительная информация:

- Секретный ключ для каждого ключа из набора (в ЭБКС)
- Значение открытого ключа для каждого ключа из набора (после генерации ключа)

3.6. Ключ УЦ для подписи сертификатов

Контейнер с назначением $K_{CA.CS}$ используется для формирования сертификатов в том случае, если криптосервис работает в составе УЦ. Для использования функций УЦ (функция **AdminService.GenCert**) необходимо предварительно создать ключевой контейнер $K_{CA.CS}$, сгенерировать версию ключа и сгенерировать или загрузить самоподписанный сертификат (см. функцию **AdminService.GenCert, AdminService.LoadCert**). При загрузке внешнего сертификата необходимо загрузить всю цепочку сертификатов, ключей, на которых подписан $K_{CA.CS}$. Это можно сделать через механизм точек доверия (см. 6.1)

Для каждой версии ключа хранится следующая дополнительная информация:

- Секретный ключ (в ЭБКС)
- Значение открытого ключа (после генерации ключа)
- Самоподписанный сертификат (после вызова функции **AdminService.GenCert, AdminService.LoadCert**)

3.7. Ключ обмена ключами ZMK

Контейнер с назначением $K_{EXCH.CS}$ необходим для осуществления обмена ключами ZMK между участниками обмена. С помощью этого ключа формируется криптоконверт со значением ключа ZMK, расшифровать который может только другая сторона. Процедура синхронизации ключа ZMK описана в 9.1.1

После генерации ключа можно сгенерировать сертификат одним из двух способов:

1. Генерация сторонним УЦ - необходимо сгенерировать запрос на сертификат (**AdminService.GenCertRequest**). После генерации сертификата сторонним УЦ на

основе запроса, нужно загрузить сертификат в криптосервис функцией (**AdminService.LoadCert**).

2. Генерация встроенным УЦ. (Для данного варианта необходимо предварительно сгенерировать ключ и сертификат $K_{CA.CS}$ (см. 3.6)) - необходимо вызвать функцию **GenCert**.

Для каждой версии ключа хранится следующая дополнительная информация:

- Секретный ключ (в ЭБКС)
- Значение открытого ключа (после генерации ключа)
- Запрос на сертификат для ключа (после вызова функции **AdminService.GenCertRequest**) или сертификат для ключа (после вызова функции **AdminService.LoadCert**)

3.8. Управление ключами криптосервиса

Процедуры управления ключами криптосервиса - регламентные процедуры, частота которых определяется как правилами пользования криптосервисом, так и бизнес-логикой приложения.

3.8.1.1. Генерация нового ключа

Процедура генерации новых ключей используемых ключевых контейнеров имеет следующие ограничения:

- Новый ключ должен быть сгенерирован ***раньше*** истечения срока службы предыдущего. Срок службы ключей каждого типа определен в правилах пользования криптосервисом.
- Технические требования проекта могут предписывать генерировать новые ключи чаще, чем требуют правила пользования (например, для того, чтобы успеть распространить ключ новой версии до истечения срока предыдущей версии).

Для генерации нового ключа криптосервиса используется функция **AdminService.GenerateServerKey**)

При генерации нового ключа криптосервиса прежние ключи в контейнере остаются активны. Для отзыва устаревших ключей криптосервиса необходимо провести процедуру отзыва (Вызов функции **AdminService.RevokeServerKey**)

3.8.1.2. Вывод ключа из использования

Процедура вывода ключа из использования имеет следующие ограничения:

- Ключ должен быть выведен из использования ***раньше*** истечения его срока службы. Срок службы ключей каждого типа определен в правилах пользования криптосервиса.
- Бизнес-логика приложения может предписывать выводить ключ из использования раньше истечения срока службы (например, если устройство, использующее ключ, выходит на связь раз в месяц, и нужно заранее обновить его ключ).

Вывод ключа из использования также необходим в случае компрометации текущей версии ключа.

Возможны 2 способа вывода ключа из использования на клиентах:

- проведение процедуры переперсонализации криптомодулей клиентов для смены ключа (см. 8).
- проведение процедуры удаленного отзыва ключа клиента (Вызов функции **AdminService.RevokeServerKey**)

После завершения процедуры вывода из использования устаревшего ключа рекомендуется удалять неиспользуемый ключ из криптосервиса для освобождения ресурсов ЭБКС и криптосервиса

с помощью функции **AdminService.RemoveServerKey**, хотя данная процедура не является обязательной. Ключ не может быть использован после выполнения процедуры вывода из использования.

Процедура вывода из использования устаревшей версии ключа криптосервиса следующая:

1. Инициация отзыва ключа криптосервиса (Вызов функции **AdminService.RevokeServerKey**)
2. Ожидание завершения отзыва ключа на всех клиентах, использующих данный ключ (Пользователь получит уведомление **AdminFeedback.ServerKeyOnClientChanged**)

4. КЛИЕНТЫ

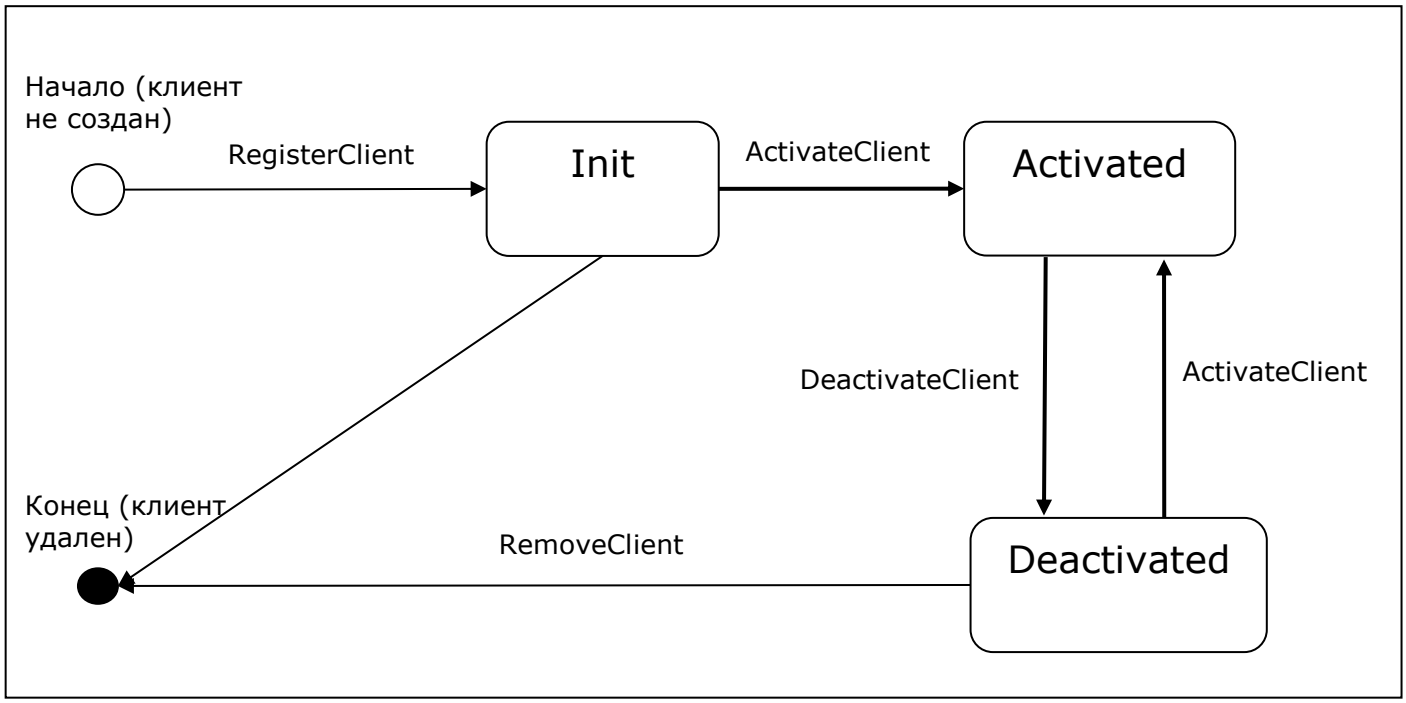
Криптосервис позволяет одновременно работать с множеством клиентов. Максимальное количество клиентов и максимальное количество одновременно активных клиентов не ограничивается криптосервисом, но может быть ограничено ресурсами криптосервера и ЭБКС.

Каждый клиент имеет уникальный идентификатор (**`ClientID`**). Каждый клиент хранит в своем ЭБ ключи для взаимодействия с криптосервисом (см. 4.1.2).

4.1.1. Жизненный цикл клиента

С точки зрения криптосервиса жизненный цикл клиента выглядит следующим образом:

Рисунок 2. Жизненный цикл клиента.



4.1.2. Ключи клиента

Для удовлетворения требований целостности, аутентичности, конфиденциальности и неотказуемости при передаче данных между клиентами и криптосервером клиент (посредством ЭБ) хранит и использует следующие криптографические ключи:

Таблица 3. Назначение ключей клиента

Назначение	Описание	Тип ключа (С–секретный ключ, П–ключевая пара, О-открытый ключ)
K_{BASE}	Базовый ключ клиента для протокола Crisp	С
K_{ADM}	Ключ администрирования клиента	С
$K_{DS.CC}$	Ключ клиента для ЭП	П
$K_{KM.CC}$	Ключ клиента для управления ключами	П
$K_{KM.CS.PUB}$	Открытая компонента ключа сервера для управления ключами	О

4.1.2.1. Ключ K_{BASE}

Ключ клиента с назначением K_{BASE} используется для обмена сообщениями с клиентом по протоколу Crisp (см. **RadioService.ReceiveMsg, AppService.PostData, AppService.SendData** и т.д.).

Для использования ключа необходимо загрузить его на ЭБ клиента при персонализации (см. **8**).

Сменить текущий ключ клиента можно через процедуру удаленного управления ключами (функция **AdminService.ChangeClientWorkKey** с указанием назначения K_{BASE}) или **AdminService.RevokeServerKey** (с указанием нового мастер-ключа MK_{BASE}).

4.1.2.2. Ключ K_{ADM}

Ключ клиента с назначением K_{ADM} используется для администрирования клиента (см. [9]).

Для использования ключа необходимо загрузить его на ЭБ клиента при персонализации (см. **8**).

Сменить текущий ключ клиента можно через процедуру удаленного управления ключами (функция **AdminService.ChangeClientWorkKey** с указанием назначения K_{ADM}) или **AdminService.RevokeServerKey** (с указанием нового мастер-ключа MK_{ADM}).

4.1.2.3. Ключ клиента для ЭП

Ключ с назначением $K_{DS.CC}$ используется для генерации ЭП клиентом. ЭП, сгенерированная клиентом, передается в составе сообщения (см. **RadioService.ReceiveMsg**), и пересылается криптосервисом вместе с данными в приложение (см. **AppFeedback.ReceiveData**).

Для использования ключа необходимо сгенерировать ключ одним из следующих способов:

- сгенерировать ключ $K_{DS.CC}$ на клиенте при персонализации (см. [6]) и загрузить его в криптосервис с помощью функции **AdminService.LoadClientPublicKey**
- сгенерировать ключ $K_{DS.CC}$ на клиенте при помощи сервиса удаленного управления ключами (функция **AdminService.GenerateClientKey**)

После завершения генерации ключа криптосервис сохранит полученную от клиента информацию об открытом ключе (его можно запросить в любое время функцией **AdminService.GetClientInfo**). Функция **AdminService.GenerateClientKey** имеет флаг, позволяющий получить запрос на сертификат ключа $K_{DS.CS}$ сразу после генерации ключа.

Сертификат ключа $K_{DS.CC}$ можно сгенерировать одним из двух способов:

1. Генерация сторонним УЦ - необходимо получить от криптосерфиса запрос на сертификат (**AdminService.GetClientCertRequest**). После генерации сертификата сторонним УЦ на основе запроса, нужно загрузить сертификат в криптосервис функцией (**AdminService.LoadClientCert**). При этом запрос на сертификат в криптосервисе заменится на сертификат (т.к. хранить и то и другое не имеет смысла).
2. Генерация встроенным УЦ. (Для данного варианта необходимо предварительно сгенерировать ключ и сертификат $K_{CA.CS}$ (см. 3.6)) - необходимо вызвать функцию **GenClientCert**.

Для каждой версии ключа хранится следующая дополнительная информация:

- Открытый ключ в сыром виде (после генерации ключа)

- Запрос на сертификат (если при генерации ключа в функции **AdminService.**GenerateClientKey был взведен флаг генерации запроса на сертификат gen_cert_req)
- сертификат (если была вызвана функция **AdminService.LoadClientCert**)

4.1.2.4. Ключ клиента для управления ключами

Контейнер с назначением $K_{KM.CC}$ используется для защиты ключевой информации при удаленном управлении ключами клиента K_{BASE} и K_{ADM} (см. функции **AdminService.ChangeClientWorkKey**, **AdminService.RevokeServerKey**).

Для использования ключа необходимо сгенерировать его одним из следующих способов:

- сгенерировать ключ $K_{KM.CC}$ на клиенте при персонализации (см. [6]) и загрузить его в криптосервис с помощью функции **AdminService.LoadClientPublicKey**
- сгенерировать ключ $K_{KM.CC}$ на клиенте при помощи сервиса удаленного управления ключами (функция **AdminService.GenerateClientKey**)

После завершения генерации ключа криптосервис сохранит полученную от клиента информацию об открытом ключе (его можно запросить в любое время функцией **AdminService.GetClientInfo**).

Для каждой версии ключа хранится следующая дополнительная информация:

- Открытый ключ в сыром виде (после генерации ключа)

Для ключей $K_{KM.CC}$, также как и для $K_{DS.CC}$, могут быть сгенерированы запросы на сертификат и сертификаты. Однако, с практической точки зрения, эту функциональность нет смысла использовать, т.к. ключ $K_{KM.CC}$ не предназначен для проверки ЭП извне, а является техническим внутренним ключом криптосервиса.

4.1.2.5. Открытая компонента ключа сервера для управления ключами

Для выполнения функций обмена ключами с криптосервисом на ЭБ клиента помимо ключа $K_{KM.CC}$ должна быть загружена открытая компонента ключа криптосервиса $K_{KM.CS}$.

Ее можно загрузить на ЭБ двумя способами:

- Записать компоненту при персонализации (см. [6])
- Загрузить/сменить компоненту удаленно при помощи функции **AdminService.ChangeServerKeyForClient**.

4.1.3. Удаленное управление ключами клиента

Замена ключей клиента - регламентная процедура, частота которой определяется как правилами пользования ЭБ клиента, так и бизнес-логикой приложения.

Новая версия ключа должна быть сгенерирована ***раньше*** истечения срока службы предыдущей версии. Срок службы ключей каждого типа определен в правилах пользования ЭБ клиента.

Все операции по удаленному управлению ключами клиента производятся после установления синхронной сессии (подробное описание механизма синхронной сессии см.[7]). Поэтому при вызове любой функции управления ключами клиента в буфере криптосервиса формируется задание на смену ключа, которое будет отправлено только после открытия клиентом синхронной сессии.

4.1.3.1. Смена рабочего ключа клиента (K_{BASE}, K_{ADM})

Данная процедура предназначена для смены ключа на одном клиенте.

Для выполнения данной функции необходимо выполнение следующих предусловий:

- Ключ $K_{KM.CS}$ сгенерирован на криптосервере и его открытая компонента загружена на ЭБ клиента (см. 4.1.3.4)
- Ключ $K_{KM.CC}$ сгенерирован на клиенте и его открытая компонента загружена на криптосервер

Задание инициируется вызовом функции **AdminService.ChangeClientWorkKey**.

При успешном завершении задачи отправляется уведомление **AdminFeedback.ClientWorkKeyChanged** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FINISHED. В этом случае криптосервис гарантирует, что ключ сменен на клиенте.

Задача смены ключа клиента является критической для выполнения, поэтому при ошибке выполнения задачи, связанной с обрывом связи, криптосервис будет пытаться повторно выполнить задачу. Если попытки не увенчались успехом, то криптосервис отправляет уведомление **AdminFeedback.EventNotification** со с типом события TASK_STOPPED. Выполнение задачи в этом случае будет возобновлено во время следующей синхронной сессии. В конечном итоге, криптосервис гарантирует, что ключ будет сменен на клиенте.

При прочих ошибках выполнения задачи отправляется уведомление **AdminFeedback.ClientTaskFailed** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FAILED. В этом случае криптосервис гарантирует, что ключ не был заменен на клиенте

4.1.3.2. Отзыв мастер-ключа (MK_{BASE}, MK_{ADM})

Данная процедура предназначена для отзыва ключей MK_{BASE}/MK_{ADM}, приведет к смене ключей K_{BASE}/K_{ADM} на ЭБ всех клиентов, имеющих K_{BASE}/K_{ADM}, выведенный из данного MK_{BASE}/MK_{ADM}.

Для выполнения данной функции необходимо выполнение следующих предусловий:

- Ключ $K_{KM.CS}$ сгенерирован на криптосервере и его открытая компонента загружена на ЭБ всех клиентов, имеющих K_{BASE}/K_{ADM}, выведенный из данного MK_{BASE}/MK_{ADM} (см. 4.1.3.4)
- Ключ $K_{KM.CC}$ сгенерирован на всех клиентах, имеющих K_{BASE}/K_{ADM}, выведенный из данного MK_{BASE}/MK_{ADM} и его открытая компонента загружена на криптосервер

Задание инициируется вызовом функции **AdminService.RevokeServerKey**

При успешном завершении задачи для каждого клиента отправляется уведомление **AdminFeedback. ClientWorkKeyChanged** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FINISHED. В этом случае криптосервис гарантирует, что ключ сменен на клиенте.

Задача смены ключа клиента является критической для выполнения, поэтому при ошибке выполнения задачи, связанной с обрывом связи, криптосервис будет пытаться повторно выполнить задачу. Если попытки не увенчались успехом, то для каждого такого клиента криптосервис отправляет уведомление **AdminFeedback.EventNotification** с типом события TASK_STOPPED. Выполнение задачи в этом случае будет возобновлено во время следующей синхронной сессии. В конечном итоге, криптосервис гарантирует, что ключ будет сменен на клиенте.

При прочих ошибках выполнения задачи для каждого клиента отправляется уведомление **AdminFeedback.ClientTaskFailed** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FAILED. В этом случае криптосервис гарантирует, что ключ не был заменен на клиенте.

4.1.3.3. Генерация асимметричного ключа клиента ($K_{KM.CC}$,$K_{DS.CC}$)

Задание инициируется вызовом функции **AdminService.GenerateClientKey** (для смены ключа на определенном клиенте)

При успешном завершении задачи отправляется уведомление **AdminFeedback.ClientKeyGenerated** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FINISHED. В этом случае криптосервис гарантирует, что ключ сменен на клиенте.

Задача генерации ключа клиента является критической для выполнения, поэтому при ошибке выполнения задачи, связанной с обрывом связи, криптосервис будет пытаться повторно выполнить задачу. Если попытки не увенчались успехом, то криптосервис отправляет уведомление **AdminFeedback.EventNotification** с типом события TASK_STOPPED. Выполнение задачи в этом случае будет возобновлено во время следующей синхронной сессии. В конечном итоге криптосервис гарантирует, что ключ будет сгенерирован на клиенте.

При прочих ошибках выполнения задачи отправляется уведомление **AdminFeedback.ClientTaskFailed** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FAILED. В этом случае криптосервис гарантирует, что ключ не был сгенерирован на клиенте

4.1.3.4. Смена открытой компоненты ключа сервера ($K_{KM.CS}$)

Задание инициируется вызовом функции **AdminService.ChangeServerKeyForClient** (для смены ключа на определенном клиенте) или **AdminService.RevokeServerKey** (для отзыва ключей $K_{KM.CS}$ на всех клиентах)

При успешном завершении задачи отправляется уведомление **AdminFeedback.ServerKeyOnClientChanged** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FINISHED. В этом случае криптосервис гарантирует, что ключ сменен на клиенте.

Задача смены ключа сервера на клиенте является критической для выполнения, поэтому при ошибке выполнения задачи, связанной с обрывом связи, криптосервис будет пытаться повторно выполнить задачу. Если попытки не увенчались успехом, то криптосервис отправляет уведомление **AdminFeedback.EventNotification** с типом события TASK_STOPPED. Выполнение задачи в этом случае будет возобновлено во время следующей синхронной сессии. В конечном итоге, криптосервис гарантирует, что ключ будет сгенерирован на клиенте.

При прочих ошибках выполнения задачи отправляется уведомление **AdminFeedback.ClientTaskFailed** и уведомление **AdminFeedback.EventNotification** с типом события TASK_FAILED. В этом случае криптосервис гарантирует, что ключ не был сгенерирован на клиенте

5. ОБМЕН СООБЩЕНИЯМИ

Криптосервис выполняет следующие функции обмена сообщениями между клиентом и приложением:

- Расшифрование входящих сообщений от клиентов и извлечение пользовательских данных
- Формирование исходящих сообщений для отправки данных от приложения к клиентам

Для использования указанного функционала необходимо подключиться к криптосервису по следующим интерфейсам:

- RadioApi для взаимодействия с клиентами по радиоканалу (По интерфейсам RadioService, RadioFeedback)
- AppApi для взаимодействия с приложением (По интерфейсам AppService, AppFeedback)

Рисунок 3. Пример схемы взаимодействия компонентов криптосервера

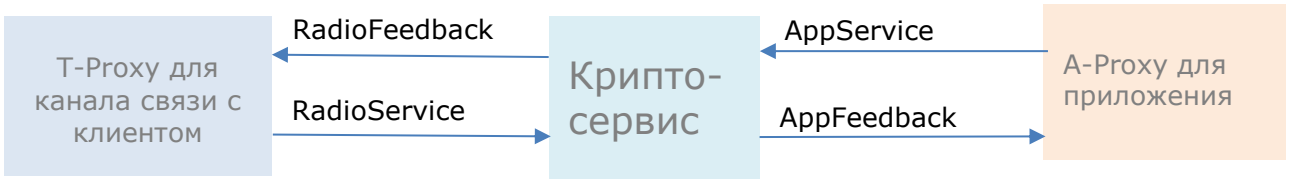


Рисунок 3 иллюстрирует минимальную конфигурацию криптосервера для возможности обмена сообщениями с клиентами.

T-Proxy для канала связи с клиентом должен быть готов в любой момент времени принять вызов функции **`RadioFeedback.SendMsg`** и инициировать при этом отправку сообщения в канал связи. Как правило, имеет смысл незамедлительно отправлять сообщение в канал связи, однако допускаются реализации API Proxy с буферизацией сообщений.

A-Proxy для приложения должен быть готов в любой момент времени принять вызов функции **`AppFeedback.ReceiveData`** и инициировать при этом отправку сообщения в сервер приложений**.** Как правило, имеет смысл незамедлительно отправлять данные приложению, однако допускаются реализации API Proxy с буферизацией сообщений.

5.1. Отправка данных от приложения без гарантии доставки

Отправка данных от приложения клиенту без гарантии доставки инициируется вызовом функции **`AppService.SendData`**. При вызове этой функции данные незамедлительно отправляются клиенту (будет вызвана функция **`RadioFeedback.SendMsg`**, которой будут переданы данные). ЭБ, обработав такое сообщение, не генерирует подтверждения доставки, а криптосервис его не ожидает.

Криптосервис не ограничивает количество и регламент отправки сообщения без гарантии доставки - они могут быть отправлены в любое время в любом количестве, независимо от состояния клиента. Ограничения могут накладываться только каналом связи (например, если клиент недоступен в данный момент, и сообщение буферизуется в сети, то буфер может переполнится). При этом, принимающая сторона будет принимать только сообщения со счетчиком большим, чем последнее принятое. Это означает, в частности, что если сообщения 1 и 2 в канале перепутались, и первым пришло сформированное позднее (2), то сообщение 1 будет отвергнуто.

5.2. Отправка данных от приложения с гарантией доставки

Отправка данных от приложения клиенту с гарантией доставки инициируется вызовом функции **`AppService.PostData`**.

Криптосервис отправляет сообщения с гарантией доставки только в рамках синхронной сессии (подробности см.[7]). До открытия синхронной сессии все данные буферизуются в криптосервисе

При успешном завершении задачи отправляется уведомление **AppFeedback.DataStateChanged** (со статусом **DATA_DELIVERED**) и уведомление **AdminFeedback.EventNotification** с типом события TASK_FINISHED. В этом случае криптосервис гарантирует что данные были успешно доставлены клиенту.

При любой ошибке выполнения задачи отправляется уведомление **AppFeedback.DataStateChanged** (со статусом **DATA_DELIVERY_FAILED**) и уведомление **AdminFeedback.EventNotification** с типом события TASK_FAILED. Попытки заново провести передачу данных выполнено не будет. При необходимости задание на передачу данных может быть заново добавлено в очередь криптосервиса. В этом случае криптосервис не дает гарантий доставки данных – они могли быть как доставлены (если уведомление о доставке не дошло) или не доставлены (если сообщение не дошло).

5.3. Прием сообщения от клиента

Входящее сообщение от клиента приходит в первую очередь в API Proxy для канала связи.

API Proxy отправляет сообщение в криптосервис с помощью **RadioService.ReceiveMsg**.

Криптосервис либо обрабатывает сообщение самостоятельно (если это служебное сообщение), либо отправляет полученные данные в API Proxy для приложения (вызов функции **AppFeedback.ReceiveData**).

6. PKI

Для возможности интеграции криптосервиса в инфраструктуру PKI, криптосервис поддерживает ряд функций для работы с сертификатами X509 для ключей $K_{DS.CS}$ и $K_{DS.CC}$, $K_{CA.CS}$ и $K_{EXCH.CS}$.

Криптосервис позволяет:

- Генерировать запросы на сертификаты для всех перечисленных типов ключей (**AdminService.GenCertRequest** для серверных ключей, для ключа $K_{KM.CC}$ запрос генерируется на ЭБ клиента и передается криптосервису автоматически при удаленной генерации нового ключа функцией **AdminService.GenerateClientKey**)
- Выгружать запросы на сертификаты для передачи сторонним УЦ (**AdminService.GetPublicKeyInfo**,**AdminService.GetClientCertRequest**)
- Загружать сертификаты, подписанные сторонним УЦ, для хранения и выдачи в информационных функциях (**AdminService.LoadCert**, **AdminService.LoadClientCert**)
- Выгружать сертификаты, предварительно сгенерированные или загруженные. Выгрузка доступна из интерфейса администрирования (**AdminService.GetPublicKeyInfo**, **AdminService.GenClientCert**), и из интерфейса для приложения (**AppService.GetCertificate**)
- Генерировать самоподписанный сертификат встроенного УЦ для $K_{CA.CS}$ (**AdminService.GenCert**)
- Генерировать и подписывать сертификаты с использованием встроенного УЦ для ключей $K_{DS.CS}$ и $K_{DS.CC}$ и $K_{EXCH.CS}$. (**AdminService.GenCert**, **AdminService.GenClientCert**)

Полный список функций приведен в 13

6.1. Точки доверия

При загрузке сертификатов, подписанных сторонним УЦ, криптосервис проверяет, что сертификат был подписан действительно доверенным УЦ. Для этого используется механизм точек доверия.

Для успешной проверки сертификата, подписанного сторонним УЦ, необходимо:

1. Загрузить корневую точку доверия - самоподписанный сертификат стороннего УЦ (**AdminService.LoadTrustPoint**)
2. Загрузить промежуточные точки доверия - сертификаты в порядке от вышестоящего к нижестоящему (т.е. чтобы при загрузке сертификата можно быть проверить его подпись) (**AdminService.LoadTrustPoint**)

После этих действий криптосервис сможет загрузить сертификаты ключей $K_{DS.CS}$, $K_{DS.CC}$, $K_{EXCH.CS}$, $K_{CA.CS}$.

Удалить просроченные или ненужные точки доверия можно с помощью функции **AdminService.RemoveTrustPoint**.

7. УПРАВЛЕНИЕ ДОСТУПОМ

Управление доступом к функциям криптосервиса зависит от типа функции. Все функции криптосервиса можно разделить на 3 группы:

- Криптографические функции – функции, работающие с ключевой информацией, и обращающиеся для этого к модулю безопасности. Для данного типа функций предусмотрена дискреционная модель разграничения доступа. Все права доступа к функциям данного типа контролируются самим модулем безопасности.
- Административные функции – функции, не использующие ключевую информацию, и не обращающиеся к криптомодулю, но меняющие состояние криптосервиса. Для выполнения функций данного типа необходима авторизация какого-либо пользователя. При попытке вызвать административную функцию без авторизации криптосервис вернет статус ACCESS_DENIED. Все права доступа к функциям данного типа контролируются криптосервисом.
- Информационные функции – функции, не меняющие состояние криптосервиса. Доступ к ним не ограничен

Криптосервис предоставляет возможность настройки прав доступа для использования криптографических функций. Данный механизм реализован посредством дискреционной модели доступа, при которой различным пользователям системы выдаются различные права доступа. Права доступа – совокупность разрешенных пользователю видов доступа к криптосервису.

Криптосервис не предоставляет механизмов для разграничения доступа к административным и информационным функциям (кроме общей проверки авторизации пользователя в административных функциях, сделанной во избежание случайной модификации данных криптосервиса). При необходимости, разграничение может быть организовано средой функционирования на уровне интерфейсов или на уровне функций.

Возможные виды доступа приведены в Таблица 4

Таблица 4. Виды доступа криптосервиса

Виды доступа	Обозначение	Описание
Обработка сообщений Crisp	AR_WORK	Шифрование/расшифрование пакетов Crisp, вычисление ЭП криптосервисом (этот вид доступа открыт всегда).
Управление ключами клиентов	AR_KM	Выполнение отложенных заданий по управлению ключами на клиентах
Персонализация	AR_PERS	Функции персонализации клиентского криптомодуля
Экспорт/импорт данных криптосервиса	AR_EXCH	Экспорт/импорт информации (в т.ч. ключевой) из/в криптосервис
Управление зональными ключами	AR_ZMK	Генерация, экспорт/импорт зональных ключей
Функции УЦ	AR_CA	Подпись сертификатов клиентов и криптосервиса
Управление ключами PKI	AR_PKI	Генерация сертификата УЦ
Управление полномочиями ЭБКС	AR_ADM	Управление пользователями криптосервиса

Для управления пользователями криптосервиса используются функции **AdminService.AddUser, AdminService.ChangeUserPassword, AdminService.ChangeUserAuthority, AdminService.DeleteUser**

Для каждого пользователя криптосервиса определены имя, пароль и права доступа. При начале работы с криптосервисом на нем создан единственный пользователь, обладающий следующими характеристиками:

Имя	admin
Пароль	admin
Права доступа	все

Права пользователя подтверждаются посредством функции **AdminService.Login**. Если права доступа не позволяют текущему пользователю выполнить операцию, то будет возвращен ошибочный статус ACCESS_DENIED (определен для каждой функции индивидуально).

Для отложенных заданий (см. 4.1.2) права доступа также проверяются отложено в момент исполнения задания. В этом случае функция инициации отложенного задания вернет корректный статус возврата, но при попытке выполнить задания криптосервер отправит уведомление **AdminFeedback.ClientTaskFailed** со статусом ACCESS_DENIED.

В Таблица 5 приведен список функций криптосервиса, и необходимых прав доступа.

Если в графе «отложенная проверка» отмечен плюс, то, поскольку операция является отложенной, права доступа будут требоваться не в момент вызова функции, а в момент выполнения отложенного задания, т.е. после начала синхронной сессии с клиентом.

Для административных функций в графе «Права доступа» указано «Авторизация»

Для информационных функций в графе «Права доступа» указано «Свободно»

Таблица 5. Права доступа для функций криптосервиса

Функция	Права доступа	Отложенная проверка
Функции интерфейса приложения		
AppService.PostData	AR_WORK	+
AppService.SendData	AR_WORK	
AppService.GetCertificate	Свободно	
AppService.GetPublicKey	Свободно	
AppService.GetClientCertificate	Свободно	
AppService.GetClientPublicKey	Свободно	
Функции интерфейса канала связи		
RadioService.ReceiveMsg	AR_WORK	
Функции управления пользователями		
AdminService.Login	Свободно	
AdminService.Logout	Свободно	
AdminService.AddUser	AR_ADM	
AdminService.ChangeUserPassword	AR_ADM	
AdminService.ChangeUserAuthority	AR_ADM	
AdminService.RemoveUser	AR_ADM	
AdminService.GetUserInfo	Свободно	
AdminService.GetAllUsers	Свободно	
Функции управления клиентами		
AdminService.RegisterClient	AR_KM	
AdminService.RegisterClients	AR_KM	
AdminService.SetClientDescription	Авторизация	
AdminService.ActivateClient	Авторизация	
AdminService.DeactivateClient	Авторизация	
AdminService.RemoveClient	Авторизация	
AdminService.GenerateClientKey	Авторизация	+
AdminService.ChangeClientWorkKey	AR_KM	+
AdminService.ChangeServerKeyForClient	AR_KM	+
AdminService.GetClientInfo	Свободно	
AdminService.GetClientsRange	Свободно	
AdminService.GetClientsInfo	Свободно	
Функции управления ключевыми контейнерами сервера		

AdminService.CreateKeyContainer	Авторизация	
AdminService.SetContainerDescription	Авторизация	
AdminService.RemoveKeyContainer	Авторизация	
AdminService.GetKeyContainerInfo	Свободно	
AdminService.GetKeyContainersInfo	Свободно	
AdminService.GetKeyContainerKeys	Свободно	
AdminService.GetKeyContainerCurrentKey	Свободно	
Функции управления ключами сервера		
AdminService.GenerateServerKey	AR_KM	
AdminService.SetKeyDescription	Авторизация	
AdminService.RevokeServerKey	AR_KM	+
AdminService.DisableServerKey	Авторизация	
AdminService.RemoveServerKey	AR_KM	
AdminService.GetServerKeyInfo	Свободно	
Функции управления сертификатами		
AdminService.GenCertRequest	AR_KM	
AdminService.LoadCert	AR_KM	
AdminService.GenCert	AR_KM	
AdminService.GetPublicKeyInfo	Свободно	
AdminService.GetCertInfo	Свободно	
AdminService.GetClientCertRequest	Свободно	
AdminService.LoadClientCert	AR_KM	
AdminService.GenClientCert	AR_KM	
AdminService.GetClientPublicKeyInfo	Свободно	
AdminService.GetClientCertInfo	Свободно	
Функции работы с точками доверия		
AdminService.LoadTrustPoint	AR_KM	
AdminService.GetTrustPointCert	Свободно	
AdminService.RemoveTrustPoint	Авторизация	
AdminService.GetTrustPointsInfo	Свободно	
Функции персонализации		
AdminService.LoadClientPublicKey	Авторизация	
AdminService.GetClientParams	Свободно	
AdminService.GetWorkKeyValue	AR_PERS	
Функции управления отложенными задачами и синхронными сессиями		
AdminService.GetClientTasksInfo	Свободно	
AdminService.CancelClientTask	Авторизация	
AdminService.FinishSyncSession	Авторизация	
Функции управления событиями		
AdminService.GetEvents	Свободно	
AdminService.RemoveEvent	Авторизация	
Функции экспорта/импорта для обмена с другими подсистемами		
AdminService.ExportZmk	AR_EXCH	
AdminService.ImportZmk	AR_EXCH	
AdminService.ExportMasterKey	AR_EXCH	
AdminService.ImportMasterKey	AR_EXCH	
API для работы с проблемами криптосервиса		
AdminService.GetServerIssues	Свободно	
Функции управления МБКС		
AdminService.GetSMInfo	Свободно	
AdminService.RepairSM	Свободно	
AdminService.DisableSE	AR_ADM	
Функции мониторинга		
AdminService.GetResourcesInfo	Свободно	
AdminService.GetLoadingInfo	Свободно	
AdminService.GetTelemetry	Свободно	
AdminService.GetClientActivityInfo	Свободно	
AdminService.GetServiceInfo	Свободно	

Api для настройки криптосервиса		
AdminService.GetSettings	Свободно	
AdminService.SetSettings	Авторизация	
AdminService.SetStepMode	Авторизация	
AdminService.SetCryptoServiceName	Авторизация	
AdminService.GetCryptoServiceName	Свободно	
AdminService.FinishCryptoService	Авторизация	
AdminService.RestartCryptoService	Авторизация	

8. ПОДГОТОВКА ЭБ КЛИЕНТА К ЭКСПЛУАТАЦИИ

В зависимости от архитектуры конкретного внедрения ПАК «Звезда», возможно несколько сценариев подготовки клиента к эксплуатации. В этом разделе приведены основные из них.

8.1.1. Подготовка ЭБ клиента без использования криптосервиса

В данном сценарии предполагается, что АРМ персонализации не имеет в своем составе криптосервис.

Данный сценарий имеет следующие особенности:

- Открытая компонента ключа $K_{KM.CS}$ загружается в ЭБ при персонализации.
- Ключ $K_{KM.CC}$ генерируется в ЭБ при персонализации и его открытая компонента передается в криптосервер по другому защищённому каналу
- Все остальные ключи необходимые клиенту для работы (рабочие ключи K_{BASE} и K_{ADM} и ключ подписи K_{DS}) загружаются/генерируются удаленно после активации клиента на сервере.

Данный сценарий предполагает, что криптосервер и АРМ персонализации предварительно вырабатывают общий ключ ZMK (согласно 9.1.1).

Криптосервер предварительно передает в АРМ персонализации следующие данные:

- Значение ключа $K_{BASE.TK}$ (зашифрованное на ZMK)
- Значение ключа $K_{ADM.TK}$ (зашифрованное на ZMK)
- Открытую компоненту ключа ($K_{KM.CS}$).

Транспортные ключи одинаковы для всех ЭБ (не диверсифицируются), являются временными и должны быть заменены процедурой удаленного управления ключами сразу же после активации клиента на криптосервере.

Подготовка клиента проводится в 2 этапа:

1. **Персонализация**. Для всех персонализируемых ЭБ:
   a. загрузка на ЭБ данных персонализации (Client ID, криптосхемы и т.д.)
   b. загрузка транспортного ключа $K_{BASE.TK}$ и $K_{ADM.TK}$ в ЭБ
   c. генерация на ЭБ ключа $K_{KM.CC}$
   d. загрузка открытой компоненты ключа криптосервера в ЭБ
   e. получение открытой компоненты сгенерированного ключа $K_{KM.CC}$
   f. формирование профиля клиента (включающего в себя ID клиента, используемые версии ключей $K_{ADM.TK}$ и $K_{BASE.TK}$ и открытую компоненту ключа $K_{KM.CC}$)
   g. передача подготовленных профилей персонализации клиентов криптосерверу
2. **Регистрация, активация и синхронизация ключей.** Данный этап происходит на криптосервере. Для всех персонализируемых ЭБ:
   a. регистрация клиента (Вызов функции **`AdminService.RegisterClient`**) с использованием данных переданных из персобюро
   b. сохранение открытых компонент сгенерированного ключа $K_{KM.CC}$, полученного из персобюро, в криптосервисе (вызов функции **`AdminService.LoadClientPublicKey`**)
   c. удаленная смена ключей K_{BASE} и K_{ADM} с транспортных на боевые (вызов функции **`AdminService.ChangeClientWorkKey`**)
   d. генерация ключа $K_{DS.CC}$ (вызов функции **`AdminService.GenerateClientKey`**)

8.1.2. Подготовка ЭБ клиента с использованием криптосервиса

В данном сценарии предполагается, что АРМ персонализации имеет в своем составе криптосервис и модуль безопасности.

Данный сценарий имеет следующие особенности:

- Рабочие ключи (K_{BASE}, K_{ADM}) диверсифицируются и загружаются в ЭБ клиента при персонализации.
- Все остальные ключи необходимые клиенту для работы (Ключи для Key Management ($K_{KM.CS}$, $K_{KM.CC}$) и для подписи ($K_{DS.CC}$) загружаются/генерируются удаленно после активации клиента на сервере.

Подготовка клиента проводится в 3 этапа:

1. **Регистрация** . Этап проводится на криптосервере для всех вводимых в эксплуатацию клиентов.
    a. регистрация всех подготавливаемых клиентов (вызов функции **`AdminService.RegisterClient`**)
    b. экспорт профиля персонализации всех подготавливаемых клиентов (вызов функции **`AdminService.ExportClient`**)
    c. передача подготовленных профилей персонализации клиентов в АРМ персонализации
2. **Персонализация**. Данный этап проводится на АРМ персонализации для всех персонализируемых ЭБ:
    a. импорт профиля персонализации клиента (вызов функции **`AdminService.ImportClient`**)
    b. запрос параметров персонализации клиента у криптосервера (вызов функции **`AdminService.GetClientParams`**).
    c. диверсификация и получение ключей K_{BASE} и K_{ADM} (вызов функции **`AdminService.GetWorkKeyValue`**)
    d. загрузка данных на ЭБ с использованием данных персонализации и диверсифицированных ключей, полученных на предыдущих шагах.
    e. формирование профиля эксплуатации клиента (Вызов функции **`AdminService.ExportClient`**)
    f. передача профилей эксплуатации клиентов на криптосервер
3. **Активация и синхронизация ключей.** Данный этап происходит на криптосервере для всех вводимых в эксплуатацию клиентов:
    a. импорт профиля эксплуатации клиента (Вызов функции **`AdminService.ImportClient`**)
    b. активация клиента (Вызов функции **`AdminService.ActivateClient`**)
    c. генерация асимметричных ключей клиента $K_{KM.CC}$ и $K_{DS.CC}$ (Вызов функции **`AdminService.GenerateClientKey`** для каждого типа ключа)
    d. загрузка открытой компоненты ключа сервера $K_{KM.CS}$ (Вызов функции **`AdminService. ChangeServerKeyForClient`** для клиента

После всех вышеописанных процедур клиент готов к эксплуатации.

9. ЭКСПОРТ/ИМПОРТ ДАННЫХ КРИПТОСЕРВИСА

В режиме экспорта/импорта данных криптосервис выполняет следующие функции:

- экспорт/импорт ключей ZMK для обмена с другими криптосервисами или внешнего хранения
- экспорт/импорт ключей MK_{BASE}, MK_{ADM} для последующего переноса клиенов, привязанных к этим ключам, на другой криптосервис
- экспорт/импорт данных клиента для переноса клиенте на другой криптосервис

Вышеописанные функции могут потребоваться в следующих случаях:

- Upgrade криптосервиса с сохранением МБКС.
- Перенос криптосервиса на новый компьютер с сохранением МБКС.
- Добавление/замена ЭБКС в кластере МБКС.
- Перенос ключей на другой МБКС (транспортный / персонализационный / резервный), в т.ч. в целях резервного копирования и замены МБКС.
- Перенос части клиентов на другой действующий криптосервис, в т.ч. в целях масштабирования криптосервиса путем установки дополнительных компьютеров и разделения базы данных клиентов на отдельные домены.

В начале работы необходимо авторизоваться с правами доступа AR_EXCH. В конце работы управляющий код должен закрыть сессию работы с криптосервисом.

Криптосервис обеспечивает целостность, аутентичность и конфиденциальность обмена секретными ключами с помощью механизма зональных ключей (см. 9.1.1). Защита от переповторов при обмене данными и гарантия доставки не обеспечиваются криптосервисом и должны обеспечиваться внешними средствами.

9.1.1. Зональные ключи

Для каждой пары участников обмена ключами и данными в ПАК «Звезда» необходимо произвести процедуру генерации общего ключа обмена (ZMK). С помощью данного ключа обеспечивается целостность и конфиденциальность при обмене секретными ключами (MK_{BASE}, KM_{ADM}) между данными участниками.

Генерация зональных ключей криптосервисом описана в 3.8.1.1

Зональный ключ для обмена между двумя участниками должен либо храниться внутри модуля безопасности (при наличии криптосервиса), либо его защита должна обеспечиваться внешними средствами (защищенный контур).

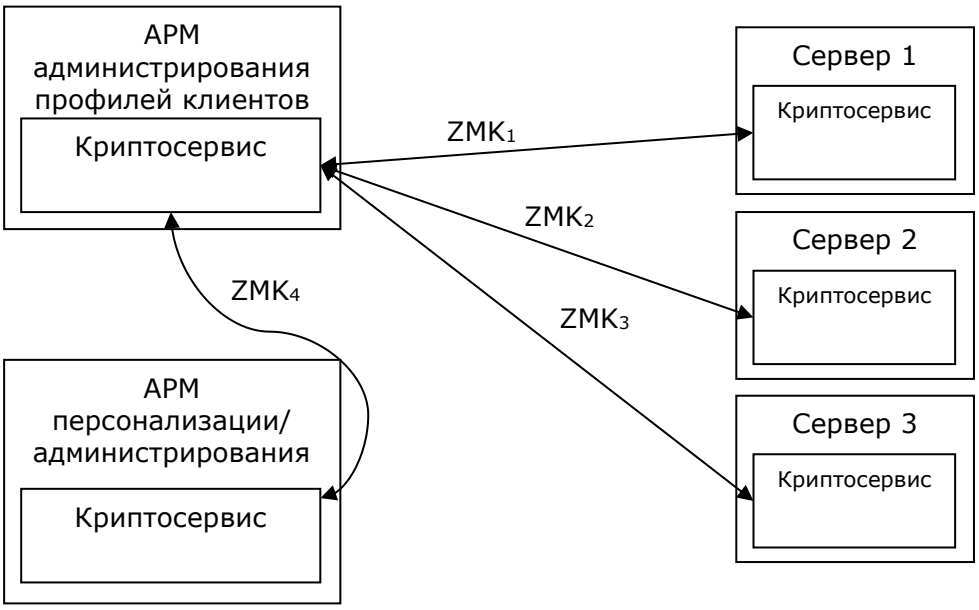
Зональный ключ используется криптосервисом для конфиденциальной передачи секретных ключей (MK_{BASE}, KM_{ADM}) при выполнении функций экспорта/импорта. В процессе экспорта участник-отправитель формирует контейнеры данных различных типов для передачи другому участнику. В процессе импорта участник-получатель принимает контейнеры, проверяет аутентичность данных, расшифровывает данные и загружает пришедшие данные.

Для использования функций экспорта данных (см. функции **`AdminService.ExportZmk`**, **`AdminService.ImportZmk`**) криптосервисом участника 1 для криптосервиса участника 2 необходимо предварительно обеспечить наличие синхронизированного зонального ключа на МБКС участника 1 и МБКС участника 2.

Синхронизация зональных ключей между двумя участниками производится посредством специального ключа $K_{EXCH.CA}$, который используется для формирования криптоконверта при передаче ключа ZMK (см. функции **`AdminService.ExportZmk`**, **`AdminService.ImportZmk`**). Для использования ключа $K_{EXCH.CS}$, в свою очередь, участники должны предварительно обменяться открытыми компонентами ключей $K_{EXCH.CA}$. Процедура обмена ключами/сертификатами ключей $K_{EXCH.CS}$ между участниками выходит за рамки настоящего технического описания.

Схема распределения зональных ключей между участниками зависит от того, какие участники должны быть связаны между собой, что определяется архитектурой конкретного внедрения ПАК "Звезда", и выходит за рамки настоящего технического описания. В частности, допускается объединение функций нескольких типов устройств в одном. Например, на сервере может одновременно запускаться и криптосервис и АРМ регистрации клиентов. Пример взаимосвязей между различными участниками приведен в Рисунок 5.

Рисунок 4. Пример взаимосвязи между участниками ПАК "Звезда"



9.1.2. Экспорт/импорт данных клиента

Процедура экспорта/импорта данных клиента может потребоваться с следующих ситуациях:

- смена управляющего сервера клиента. В этом случае необходимо удалить клиента с одного сервера и добавить на другой.

Резервное копирование данных клиентаДля проведения процедуры экспорта/импорта данных клиента необходимо предварительно согласовать ключи ZMK между сторонами обмена (см. 9.1.1)

В настоящие момент функционал экспорта/импорта данных клиента не поддерживается криптосервисом

9.1.3. Экспорт/импорт мастер-ключа криптосервиса (MK_{BASE}, MK_{ADM})

Процедура экспорта/импорта мастер-ключей криптосервиса нужна для передачи ключей MK_{BASE}, KM_{ADM} от от одного криптосервиса к другому. Эта процедура необходима в следующих ситуациях:

- перевод части клиентов на другой криптосервис.
- персонализация ранее зарегистрированных клиентов в персобюро, которое не имеет доступа к криптосервису
- резервное копирование ключа

Для проведения процедуры экспорта/импорта мастер-ключей необходимо предварительно согласовать ключи ZMK между сторонами обмена (см. 9.1.1)

Процедура экспорта/импорта ключа следующая:

1. Экспорт ключа из исходного криптосервиса (вызов функции **`AdminService.ExportMasterKey`**
2. Импорт ключа в принимающий криптосервис (вызов функции **`AdminService.ImportMasterKey`**.

9.1.4. Экспорт/импорт данных криптосервиса

Экспорт/импорт данных криптосервиса может потребоваться, в частности, при резервном копировании или при копировании данных с одного сервера на другой (для зеркалирования или формирования стека серверов для распределения нагрузки).

Для проведения процедуры экспорта/импорта данных криптосервиса необходимо предварительно согласовать ключи ZMK между сторонами обмена (см. 9.1.1)

В настоящие момент функционал экспорта/импорта данных криптосервиса не поддерживается криптосервисом.

10. НАСТРОЙКИ

Настройки криптосервиса разделены на 2 группы:

1. Статические настройки, изменяемые через файл конфигурации
2. Динамические настройки, изменяемые через интерфейс администратора (**AdminService**)

10.1. Статические настройки

Статические настройки криптосервиса конфигурируются с помощью файла настроек crypto_service.ini. После изменения файла необходимо перезапустить криптосервис.

Криптосервис имеет следующие статические настройки:

- **Настройки интерфейса криптосервиса** - TCP порты, которые криптосервис открывает для всех предоставляемых интерфейсов
- **Количество потоков обработки** - Кол-во потоков, которое будет заниматься различными задачами в криптосервисе
- **Настройки ЭБКС**
- **Настройки логирования** – уровень детализации, формат вывода

10.2. Динамические настройки

Динамические настройки конфигурируются через интерфейс администратора (**AdminService**). Динамические настройки вступают в силу сразу же после вызова функции криптосервиса.

Криптосервис имеет следующие динамические настройки:

- **Сроки службы ключей** – тот срок, после которого ключи считаются истекшими. Для корректного функционирования ПАК «Звезда» необходимо выполнить удаленную смену ключей до этого времени, либо вывести ключ из использования.
- **Кол-во дней, за которое начать предупреждать об истечении срока службы ключей** – за этот срок до истечения ключа криптосервис будет выдавать предупреждение о необходимости смены ключа через механизм **ServerIssues** (подробности см. 12.1)
- **Максимальное время ожидания ответа клиента в синхронной сессии** – по истечении этого времени синхронная сессия с клиентом будет прервана.
- **Максимальное время ожидания начала задачи после ее планирования** – по истечении этого времени запланированная задача будет отменена. Информация об этом будет отправлена криптосервисом через механизм событий (подробности см. 12.2).
- **Период фиксации в БД метрик нагрузки** – период между снятием метрик нагрузки криптосервиса (подробности см. 12.4)
- **Период автоматической проверки состояния криптосервиса** – период, через который криптосервис осуществляет проверку своего состояния. В результате проверки могут быть сгенерированы новые проблемы и события

Настроки можно изменить/запросить посредством функций **AdminService.SetSettings / AdminService.GetSettings**

11. УПРАВЛЕНИЕ МОДУЛЕМ БЕЗОПАСНОСТИ КРИПТОСЕРВЕРА

Хотя криптосервис был разработан с прицелом на максимальную абстракцию пользователя от МБКС, тем не менее, в некоторых случаях необходимо иметь возможность диагностики и управления МБКС. Это может потребоваться в следующих ситуациях:

- возникновение сбоя в работе одного из чипов.
- регламентная диагностика чипов ЭБКС.

Криптосервис предоставляет следующие возможности для управления МБКС:

- Запрос актуальной информации о модуле безопасности (**`AdminService.GetSMInfo`**)
- Диагностика/восстановление работы устройств ЭБКС в составе МБКС (**`AdminService.RepairSM`**).
- Вывод ЭБКС из использования (**`AdminService.DisableSE`**).

Чип ЭБКС может находиться в следующих возможных состояниях:

Состояние	Описание
ACTIVE	ЭБКС активен
FAILED	Имел место сбой (при старте или в процессе работы). Работа с ЭБКС временно приостановлена.
DISABLED	ЭБКС выведен из использования

При обычной работе МБКС чипы находятся в состоянии ACTIVE. При возникновении проблем с коммуникацией чип может перейти в состояние FAILED, что повлечет формирование криптосервисом проблемы DEVICE_BROKEN (Таблица 6).

В этом случае пользователь должен:

- Попытаться восстановить работу МБКС с помощью функции **`AdminService.RepairSM`**. При этом восстановленные чипы перейдут в состояние ACTIVE. Те чипы, которые не удалось восстановить, останутся в состоянии FAILED
- Запросить актуальную информацию о модуле безопасности с помощью функции **`AdminService.GetSMInfo`**. Это позволит узнать, какие из чипов не удалось восстановить.
- Если часть чипов не удалось восстановить, то нерабочие чипы следует вывести из использования с помощью функции **`AdminService.DisableSE`.** При этом указанные чипы перейдут в состояние DISABLED, и могут быть сгенерированы проблемы «Ключ потерян», «Ключ не резервирован» или «Ключ неэффективен» (см. 12.1) для тех ключей, которые хранились на данном чипе.

12. МОНИТОРИНГ

Криптосервис предоставляет следующие механизмы для мониторинга состояния:

- Список проблем
- События
- Информация о ресурсах
- Информация о нагрузке
- Информация об активности клиента
- Общая информация о криптосервисе

12.1. Список проблем

Список проблем криптосервиса формируется по запросу, и каждая из проблем списка требует решения определенным образом, в зависимости от типа проблемы. Пока проблема не решена, криптосервис не может функционировать в полноценном режиме. Проблема будет присутствовать в возвращаемом списке до тех пор, пока не будет решена.

Криптосервис предоставляет возможность запросить текущий список проблем с помощью функции **GetServerIssues**.

Список потенциальных проблем криптосервиса:

- **Истек срок службы ключа/пароля** – проблема может быть решена двумя способами – либо с помощью механизма удаленного управления ключами, либо посредством вывода проблемного ключа/пароля из использования. Пока проблема не решена, каждое использование истекшего ключа будет приводить к событию EXPIRED_KEY_USED (см. 12.2)
- **Ключ потерян** – в связи с выходом из строя МБКС, ключ физически более не доступен. Проблема решается отвязкой всех клиентов от данного ключа, или удалением соответствующих клиентов.
- **Ключ не резервирован**– в связи с выходом из строя некоторых чипов МБКС, ключ использует только один чип МБКС, и механизм резервирования не работает. Это значит, что при выходе из строя этого МБКС, содержащего значение ключа, ключ станет неработоспособным и перейдет в разряд «Потерян». Проблема решается срочной сменой ключа на всех привязанных клиентах. Проблема может формироваться только для $K_{KM.CS}$ и $K_{DS.CS}$.
- **Ключ неэффективен –** ключ использует не все чипы МБКС. Это может быть связано с тем, что после генерации ключа к МБКС были подключены новые чипы. Проблема может формироваться только для $K_{KM.CS}$ и $K_{DS.CS}$. Проблема решается перегенерацией ключа данного типа и заменой старого ключа на новый на всех клиентах.

12.2. События

События происходят в криптосервисе в определенный момент времени, фиксируются через механизм событий, и носят информативный характер. События отличаются от проблем тем, что они единовременны, и не требуют обязательных действий по администрированию криптосервиса. Криптосервис может продолжить работу.

Таблица 6. Список возможных событий

Тип	Описание	Сохранение в БД
TASK_FINISHED	Отложенное задание завершено успешно	-
SERVER_KEY_BROKEN	Ключ сервера вышел из строя	+
DEVICE_BROKEN	Чип ЭБКС вышел из строя	+
TASK_STOPPED	Отложенное задание приостановлено из-за проблемы	+
TASK_FAILED	Отложенное задание завершено с ошибкой	+
TASK_CANCELED_BY_TIMEOUT	Отложенное задание отменено. Истек таймаут ожидания	+
SYNC_SESSION_CLOSED_BY_TIMEOUT	Синхронная сессия закрыта. Истек таймаут ожидания	+
EXPIRED_KEY_USED	Использование просроченного ключа	+

При возникновении любого события криптосервис вызовет функцию **AdminFeedback.EventNotification**. Кроме того, список событий, помеченных в Таблица 6 как сохраняемые, можно запросить в любое время через функцию **AdminService.GetEvents**.

12.3. Информация о ресурсах

Криптосервис предоставляет информацию о занятых/свободных ресурсах посредством функции **AdminService.GetResourcesInfo**. Информация по ресурсам в первую очередь относится к ресурсам модуля безопасности. Ресурс модуля безопасности по хранению разного типа ключей, паролей ограничен и определяется начальными настройками модуля безопасности и количеством чипов ЭБКС в кластере. Список ресурсов, по которым можно получить информацию, см. в описании функцию **AdminService.GetResourcesInfo**

12.4. Информация о нагрузке

Информация о нагрузке на криптосервис включает такие сведения, как:

- количество обработанных команд различных типов
- объем обработанного трафика
- суммарное время выполнения команд

Криптосервис предоставляет информацию о нагрузке на криптосервис в следующем виде:

- **текущая статистика по нагрузке и операциям** – доступна через функцию **AdminService.GetLoadingInfo**
- **статистика за определенный промежуток времени** – доступна через функцию **AdminService.GetTelemetry**

Период сохранения статистики во 2-м варианте настраивается через динамические настройки криптосервиса (см. 10.2)

12.5. Информация об активности клиента

Информация об активности клиента нужна для анализа текущего состояния клиента.

Информация об активности клиента включает в себя:

- Признак выполнения синхронной сессии в настоящий момент
- Время получения от клиента последнего сообщения

Информация доступна посредством функции **AdminService.GetClientActivityInfo**

12.6. Общая информация о криптосервисе

Информация об активности криптосервиса нужна для анализа текущего состояния криптосервиса.

Общая информация о криптосервисе включает в себя:

- Время старта криптосервиса
- Время работы криптосервиса с момента старта

Информация доступна посредством функции **`AdminService.GetServiceInfo`**

13. ПРОГРАММНЫЙ ИНТЕРФЕЙС (API)

Программный интерфейс криптосервиса представлен в виде файлов в формате ZDL (описание формата см. [8] Ф). API состоит из следующих файлов:

Название	Файл	Описание
RadioService	radio_service.zdl	Интерфейс для вызова функций криптосервиса из T-Proxy
RadioFeedback	radio_feedback.zdl	Интерфейс для вызова функций T-Proxy из криптосервиса
AppService	app_service.zdl	Интерфейс для вызова функций криптосервиса из A-Proxy
AppFeedback	app_feedback.zdl	Интерфейс для вызова функций A-Proxy из криптосервиса
AdminService	admin_service.zdl	Интерфейс для вызова функций криптосервиса из АРМ администрирования
AdminFeedback	admin_feedback.zdl	Интерфейс для вызова функций АРМ администрирования из криптосервиса
AdminLib	admin_lib.zdl	Общая библиотека, в которой описаны структуры и перечисления, используемые в нескольких модулях

Для использования интерфейса необходимо скомпилировать вышеперечисленные модули для A-Proxy, T-proxy и для АРМ администрирования с TCP ролью «клиент» (подробности см. [8])

Периодически API криптосервиса обновляется, и в этом случае необходимо обновить файлы ZDL для всех пользователей (A-Proxy, T-proxy и АРМ администрирования), взаимодействующих с криптосервисом. Для минимизации возможной ошибки при подключении пользователя к криптсервису, происходит автоматическая сверка версии интерфейса (поле **`Version`** в шапке файлов .zdl должно совпадать в модулях пользователей и криптосервиса). Подробности см. [8], раздел «согласование конфигураций».

14. УСТАНОВКА КРИПТОСЕРВИСА

Дистрибутив криптосервиса включает в себя следующие компоненты:

1) arm – арм администрирования;
2) cs – криптосервис с конфигурационным файлом crypto_service.ini;
3) doc – документация, актуальная на момент сборки пакета;
4) sam – скрипты инициализации ЭБКС;
5) zdl – интерфейсы криптосервиса;
6) util – вспомогательные инструменты;

Для установки криптосервиса необходимо:

- Физически установить МБКС на сервер
- Установить драйвера PC/SC для доступа криптосервиса к МБКС
- Скопировать содержимое дистрибутива на сервер
- Инициализировать МБКС
- Настроить криптосервис через файл настроек cs /crypto_service.ini
- Запустить криптосервис

После выполнения этих шагов криптосервис готов к работе.

Подробности осуществления вышеуказанных шагов см. в инструкции по установке, поставляемой вместе с дистрибутивом.

Для перезагрузки криптосервиса можно выключить криптосервис из терминальной консоли (Ctrl+Z или крестик).

Следует обратить внимание, что в процессе работы криптосервис может создавать в директории программы (в нашем случае «cs») служебные файлы. Это могут быть, в частности, файлы базы данных. Эти файлы необходимы для корректной работы криптосервиса, и не должны модифицироваться/удаляться.

СПИСОК ИСТОЧНИКОВ

[1] Протокол защищенного обмена для индустриальных систем (CRISP 1.0). Проект. Технический комитет по стандартизации «Криптографическая защита информации»

[2] Р 50.1.113 2016. Криптографические алгоритмы, сопутствующие применению алгоритмов электронной цифровой подписи и функции хэширования

[3] ГОСТ Р 34.12 - 2015. Блочные шифры.

[4] Р50.1.114-2016 (Наборы эллиптических кривых)

[5] ПАК «Звезда». Общее техническое описание.

[6] ПАК «Звезда». Техническое описание криптомодуля.

[7] ПАК «Звезда». Протокол обмена.

[8] Технология удаленного вызова процедур

[9] Операционная система Trust 3.21D Руководство программиста

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Версия	Дата	Автор	Содержание изменения
1	06.05.2019	Сергеев В.Л.	Создание документа
2	06.06.2019	Сергеев В.Л.	Переработка документа Добавление сценариев использования
3	05.09.2019	Сергеев В.Л.	Доработка API
4	18.10.2019	Сергеев В.Л.	- Изменение API для работы с сертификатами и генерации/проверки подписи. - Более подробное описание функций API.
5	16.01.2020	Сергеев В.Л.	- Добавление параметра `has_dl_server_quota в настройки клиента` - Добавление возможности отправки Downlink данных без подтвекржения доставки - Добавление события TASK_REPORT
6	23.01.2020	Сергеев В.Л.	- Переработка формата событий --Добавление интерактивной сессии --Добавление функции запроса параметров персонализации клиента
7	30.03.2020	Сергеев В.Л.	- Добавлени раздела “Ключевая система” - Добавление раздела “Клиенты” - Добавление раздела “Подготовка клиента” - Небольшие уточнения и правки по API
7	23.04.2020	Сергеев В.Л.	- Переработка раздела “Ключевая система” - Небольшие уточнения и правки по API
8	08.05.2020	Сергеев В.Л.	- Дальнейшая доработка раздела “Ключевая система” - Убраны лишние типы событий - Небольшие уточнения и правки по API
9	01.12.2020	Сергеев В.Л.	- Дополнение глав 1,2,3 - Добавление главы 4 (Управление доступом) - Актуализация API и сценариев
10	29.10.2021	Сергеев В.Л.	Переработка документа. Добавление глав -Мониторинг -Настройка -PKI