ГОСУДАРСТВЕННЫЙ СТАНДАРТ
РЕСПУБЛИКИ БЕЛАРУСЬ

СТБ 34.101.78-2019

# Информационные технологии и безопасность
# ПРОФИЛЬ ИНФРАСТРУКТУРЫ ОТКРЫТЫХ КЛЮЧЕЙ

# Інфармацыйныя тэхналогіі і бяспека
# ПРОФІЛЬ ІНФРАСТРУКТУРЫ АДКРЫТЫХ КЛЮЧОЎ

Госстандарт
Минск

УДК 004.056.55(083.74)(476) МКС 35.240.40

**Ключевые слова**: инфраструктура открытых ключей, сертификат открытого ключа, поставщик услуг доверия, криптографический токен, ключевой контейнер

## Предисловие

Цели, основные принципы, положения по государственному регулированию и управлению в области технического нормирования и стандартизации установлены Законом Республики Беларусь «О техническом нормировании и стандартизации».

1 РАЗРАБОТАН учреждением Белорусского государственного университета «Научно-исследовательский институт прикладных проблем математики и информатики»

ВНЕСЕН Оперативно-аналитическим центром при Президенте Республики Беларусь

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ постановлением Госстандарта Республики Беларусь от 8 июля 2019 г. № 42

3 ВВЕДЕН ВПЕРВЫЕ

Издан на русском языке

# Содержание

**ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ**

**Информационные технологии и безопасность**
# ПРОФИЛЬ ИНФРАСТРУКТУРЫ ОТКРЫТЫХ КЛЮЧЕЙ

**Інфармацыйныя тэхналогіі і бяспека**
# ПРОФІЛЬ ІНФРАСТРУКТУРЫ АДКРЫТЫХ КЛЮЧОЎ

Information technology and security
A public key infrastructure profile

**Дата введения 2019-10-01**

## 1 Область применения

Настоящий стандарт устанавливает профиль инфраструктуры открытых ключей (ИОК), рекомендуемый для использования в Республике Беларусь. В стандарте определяются стороны ИОК, процессы их взаимодействия, протоколы взаимодействия, уточняются форматы объектов ИОК.

Настоящий стандарт применяется при разработке средств криптографической защиты информации и информационных систем, в которых используются открытые ключи.

## 2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие технические нормативные правовые акты в области технического нормирования и стандартизации (далее — ТНПА):

СТБ 34.101.17-2012 Информационные технологии и безопасность. Синтаксис запроса на получение сертификата

СТБ 34.101.19-2012 Информационные технологии и безопасность. Форматы сертификатов и списков отозванных сертификатов инфраструктуры открытых ключей

СТБ 34.101.21-2009 Информационные технологии. Интерфейс обмена информацией с аппаратно-программным носителем криптографической информации (токеном)

СТБ 34.101.23-2012 Информационные технологии и безопасность. Синтаксис криптографических сообщений

СТБ 34.101.26-2012 Информационные технологии и безопасность. Онлайновый протокол проверки статуса сертификата (OCSP)

СТБ 34.101.27-2011 Информационные технологии и безопасность. Требования безопасности к программным средствам криптографической защиты информации

СТБ 34.101.31-2011 Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности

СТБ 34.101.45-2013 Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых

СТБ 34.101.47-2017 Информационные технологии и безопасность. Криптографические алгоритмы генерации псевдослучайных чисел

СТБ 34.101.60-2014 Информационные технологии и безопасность. Алгоритмы разделения секрета

СТБ 34.101.65-2014 Информационные технологии и безопасность. Протокол защиты транспортного уровня (TLS)

СТБ 34.101.67-2014 Информационные технологии и безопасность. Инфраструктура атрибутных сертификатов

СТБ 34.101.77-2016 Информационные технологии и безопасность. Алгоритмы хэширования

СТБ 34.101.79-2019 Информационные технологии и безопасность. Криптографические токены

СТБ 34.101.80-2019 Информационные технологии и безопасность. Расширенные электронные цифровые подписи

СТБ 34.101.81-2019 Информационные технологии и безопасность. Протоколы службы заверения данных

СТБ 34.101.82-2019 Информационные технологии и безопасность. Протокол постановки штампа времени

ГОСТ 34.973-91 (ИСО 8824-87) Информационная технология. Взаимосвязь открытых систем. Спецификация абстрактно-синтаксической нотации версии 1 (АСН.1)

ГОСТ 34.974-91 (ИСО 8825-87) Информационная технология. Взаимосвязь открытых систем. Описание базовых правил кодирования для абстрактно-синтаксической нотации версии 1 (АСН.1)

ГОСТ 27463-87 Системы обработки информации. 7-битные кодированные наборы символов

Примечание — При пользовании настоящим стандартом целесообразно проверить действие технических нормативных правовых актов в области технического нормирования и стандартизации (далее — ТНПА) по каталогу, составленному по состоянию на 1 января текущего года, и по соответствующим информационным указателям, опубликованным в текущем году. Если ссылочные ТНПА заменены (изменены), то при пользовании настоящим стандартом следует руководствоваться действующими взамен ТНПА. Если ссылочные ТНПА отменены без замены, то положение, в котором дана ссылка на них, применяется в части, не затрагивающей эту ссылку.

## 3 Термины и определения

В настоящем стандарте применяют термины, установленные в СТБ 34.101.19, СТБ 34.101.23, СТБ 34.101.27, СТБ 34.101.45, СТБ 34.101.65, СТБ 34.101.67, СТБ 34.101.81, СТБ 34.101.82, а также следующие термины с соответствующими определениями:

**3.1 агент**: Криптографический автомат, выполняющий определенные технологические процессы по поручению другой стороны.

**3.2 защищенное соединение**: Соединение, которое обеспечивает конфиденциальность, контроль целостности и возможно подлинности сообщений.

Примечание — Контроль подлинности сообщений от стороны $A$ к стороне $B$ обеспечивается после того, как $B$ провела аутентификацию $A$.

**3.3 идентификационный атрибут**: Компонент идентификационных данных.

**3.4 идентификационные данные**: Данные, которые однозначно характеризуют определенную сторону в определенном контексте.

Примечание — В разных контекстах могут использоваться различные идентификационные данные одной и той же стороны.

**3.5 криптографический автомат**: Средство криптографической защиты информации, которое выступает от собственного лица при взаимодействии с другими сторонами и которое основное время работает автономно, без внешнего управления.

Примечание — Примерами криптографических автоматов являются IP-шифраторы, автономные терминалы, устройства Интернета вещей.

**3.6 криптографический токен**: Средство криптографической защиты информации, имеющее конкретного владельца и выступающее от его лица при взаимодействии с другими сторонами.

Примечание — В настоящем стандарте криптографический токен хранит один или несколько личных ключей владельца и реализует операции с ними. На токене могут размещаться идентификационные данные владельца.

**3.7 конвертованные данные**: Данные, защищенные на секретном ключе и сопровождаемые этим ключом, защищенном на открытом ключе получателя.

**3.8 оператор**: Физическое лицо, которое отвечает за выполнение определенных технологических процессов другой стороны.

**3.9 подписанные данные**: Данные, сопровождаемые электронной цифровой подписью отправителя.

**3.10 пользователь**: Физическое лицо.

**3.11 поставщик услуг доверия**: Сторона, которая помогает установить доверенные отношения между другими сторонами, предоставляя определенную информацию по их запросу.

Примечание — К поставщикам услуг доверия относятся удостоверяющие центры, центры атрибутных сертификатов, регистрационные центры, OCSP-серверы, службы штампов времени, заверения данных и идентификации.

**3.12 прикладная система**: Информационная система, которая использует услуги доверия, предоставляемые инфраструктурой открытых ключей.

**3.13 регистрационный центр**: Поставщик услуг доверия, который формирует идентификационные данные пользователя или проверяет и заверяет их для удостоверяющего центра.

Примечание — Регистрационный центр может дополнительно регистрировать аутентификационные данные пользователя и передавать их службе идентификации.

**3.14 служба идентификации**: Поставщик услуг доверия, который проводит аутентификацию пользователей и авторизует на доступ к их ресурсам.

**3.15 сторона**: Активный элемент (центр, сервер, служба, лицо, устройство), который является частью инфраструктуры открытых ключей или использует сервисы инфраструктуры и, как правило, располагает одним или несколькими сертификатами открытых ключей, выпущенными в инфраструктуре.

**3.16 терминал**: Сторона, которая взаимодействует с криптографическим токеном по защищенному соединению после аутентификации перед ним и, возможно, встречной аутентификации токена.

**3.17 удостоверение**: Документ на физическом носителе (бумага, пластик), выпущенный доверенной стороной и содержащий идентификационные данные пользователя или организации.

**3.18 физическое лицо**: Гражданин Республики Беларусь (резидент) или другой страны (нерезидент), субъект гражданского права.

**3.19 цепочка сертификатов**: Маршрут сертификации.

**3.20 юридический представитель**: Физическое лицо, сотрудник или представитель юридического лица.

**3.21 юридическое лицо**: Зарегистрированная в Республике Беларусь организация, субъект гражданского права.

# 4 Обозначения и сокращения

## 4.1 Обозначения

| | |
|---|---|
| $\{0,1\}^n$ | множество всех слов длины $n$ в алфавите $\{0,1\}$; |
| $\{0,1\}^*$ | множество всех слов конечной длины в алфавите $\{0,1\}$; |
| $u \parallel v$ | объединение (конкатенация) слов $u, v \in \{0,1\}^*$; |
| $\{0,1\}^{n*}$ | множество всех слов из $\{0,1\}^*$, длина которых кратна $n$; |
| (символы 0–F)$_{16}$ | представление $u \in \{0,1\}^{4*}$ шестнадцатеричным словом, при котором последовательным четырем символам $u$ соответствует один шестнадцатеричный символ (например, $10100010 = \mathtt{A2}_{16}$); |
| $\mathtt{hex}(u)$ | для $u \in \{0,1\}^{4*}$ текстовая строка, полученная заменой символов в шестнадцатеричном представлении $u$ на соответствующие печатные символы (например, $\mathtt{hex}(10100010) = \mathtt{hex}(\mathtt{A2}_{16}) = \text{'A2'}$). |

## 4.2 Сокращения

В настоящем стандарте применяют следующие сокращения:

АСН.1 — абстрактно-синтаксическая нотация версии 1 (ГОСТ 34.973);

КА — криптографический автомат;

КТ — криптографический токен;

КУЦ — корневой удостоверяющий центр;

ПУД — поставщик услуг доверия;

ПУЦ — подчиненный удостоверяющий центр;

РУЦ — республиканский удостоверяющий центр;

РЦ — регистрационный центр;

СЗД — служба заверения данных (СТБ 34.101.81);

СИ — служба идентификации;

СКЗИ — средство криптографической защиты информации;

СОК — сертификат открытого ключа (СТБ 34.101.19);

СОС — список отозванных сертификатов (СТБ 34.101.19);

СШВ — служба штампов времени (СТБ 34.101.82);

УЦ — удостоверяющий центр (СТБ 34.101.19);

ФЛ — физическое лицо;

ЦАС — центр атрибутных сертификатов (СТБ 34.101.67);

ЭД — электронный документ;

ЭК — эллиптическая кривая;

ЭЦП — электронная цифровая подпись;

ЮЛ — юридическое лицо;

ЮП — юридический представитель;

DER — отличительные правила кодирования АСН.1 (distinguished encoding rules, СТБ 34.101.19 [приложение Б]);

DNS — система доменных имен (domain name system, [1]);

HTTP — протокол передачи гипертекста (hypertext transfer protocol, [2]);

OCSP — онлайновый протокол проверки статуса сертификата (online certificate status protocol, СТБ 34.101.26);

URI — унифицированный идентификатор ресурса (uniform resource identifier, [3]).

# 5 Общие положения

## 5.1 Назначение

Для построения систем электронного документооборота, аутентификации, защищенного онлайн-взаимодействия используются криптографические алгоритмы с двумя ключами: личным и открытым. Личный ключ хранится в секрете, открытый рассылается заинтересованным сторонам. Нарушение целостности или подлинности открытого ключа при его распространении приводит к потере стойкости криптографического алгоритма с последующей компрометацией целевой системы.

ИОК предназначены для надежного распространения открытых ключей. Существует несколько подходов к построению ИОК. Основной определяется международным стандартом X.509 [4], частично введенном в СТБ 34.101.19 и поддерживаемом другими действующими ТНПА.

ТНПА определяют различные аспекты взаимодействия сторон ИОК. При этом задаваемые ТНПА решения во многих случаях являются чересчур гибкими и недостаточно конкретными. Например, стандартные объекты ИОК содержат опциональные компоненты, компоненты с выбираемым или даже открытым форматом. Для обеспечения совместимости СКЗИ, используемых в ИОК, требуется реализовывать сразу несколько вариантов форматов, и все равно совместимость зачастую не достигается.

Другая проблема — несмотря на большой охват и гибкость, ТНПА не покрывают все аспекты взаимодействия. Остаются области, которые требуют дополнительной стандартизации, например, процессы выпуска и отзыва сертификатов открытых ключей или правила транспорта данных ИОК по каналам Интернет.

Настоящий стандарт конкретизирует (профилирует) аспекты взаимодействия сторон ИОК. Конкретизация детальна настолько, что разработку СКЗИ для работы в ИОК или с ИОК можно вести, руководствуясь только настоящим стандартом, без дополнительных инструкций о правилах взаимодействия.

Прозрачность и конкретика упрощают разработку и эксплуатацию ИОК, использование сервисов ИОК в других информационных системах. Упрощается и работа экспертов, которые проводят оценку надежности подсистем ИОК.

Открытые ключи, для управления которыми создается инфраструктура, одновременно используются для организации взаимодействия ее сторон. Процессы взаимодействия базируются на криптографических алгоритмах с открытым ключом, описанных в разделе 6. Построение инфраструктуры на основе обрабатываемых в ней открытых ключей, без внешних криптографических объектов, делает ИОК самодостаточной.

## 5.2 Инфраструктура открытых ключей

Согласно X.509, открытые ключи распространяются в форме сертификатов — самодостаточных объектов, не привязанных к конкретному сетевому хранилищу. X.509 фактически задает распределенный справочник открытых ключей.

Сертификат связывает открытый ключ с идентификационными данными владельца и другими атрибутами. Связывание выполняет УЦ — специальная служба ИОК, которая подписывает данные сертификата на своем личном ключе. Подпись, а также идентифика-

ционные данные УЦ являются частями сертификата. УЦ называют эмитентом, владельца открытого ключа — субъектом. Подпись переносит доверие к эмитенту сначала на выпущенные им сертификаты, а затем и на их субъектов.

Для проверки сертификата нужно получить открытый ключ УЦ. Этот ключ также распространяется в форме сертификата, выданного другим, высшим по иерархии, УЦ. Открытый ключ этого УЦ вкладывается в сертификат, выпущенный еще одним УЦ, и так далее. В конце концов получается цепочка сертификатов, в которой каждый следующий сертификат подписывается на открытом ключе текущего.

Цепочка начинается сертификатом, который выдает корневой, высший по иерархии, УЦ. КУЦ сам подписывает свой сертификат, все стороны ИОК получают его доверенным образом. Сертификат КУЦ называется точкой доверия. Это отправная точка цепочки сертификатов к другим сторонам ИОК. Цепочка переносит доверие с КУЦ на эти стороны. В этой связи КУЦ и другие УЦ называются поставщиками услуг доверия.

Услуги доверия не исчерпываются выпуском сертификатов. Примеры других поставщиков — регистрационные центры, проверяющие корректность идентификационных данных субъектов перед выпуском их сертификатов, серверы информирования о статусе сертификатов, службы подтверждения существования данных к определенному моменту времени. ПУД и другие стороны ИОК описываются в 7.

Атрибуты, указываемые в сертификате, касаются субъекта, эмитента или собственно сертификата. Эти атрибуты могут быть вынесены в специальный отдельный сертификат, который называется атрибутным. Атрибутный сертификат содержит ссылку на сертификат открытого ключа, но не содержит сам открытый ключ. Атрибутные сертификаты выпускает ЦАС — еще один ПУД, определенный в СТБ 34.101.67. Обычно атрибуты контекстно-зависимы и поэтому весьма разнообразны. В целом управление атрибутами — это отдельная инфраструктура управления привилегиями, которая в настоящем стандарте не конкретизируется. Профилируется только сертификат открытого ключа ЦАС.

Взаимодействие между сторонами ИОК в настоящем стандарте регулируется на нескольких уровнях: форматы пересылаемых данных (раздел 8), процессы взаимодействия (раздел 9), транспорт данных (раздел 10).

Форматы данных определяются на языке АСН.1, определенном в ГОСТ 34.973. В приложении А приводится модуль АСН.1, в котором описываются специальные форматы ИОК, определяются идентификаторы объектов ИОК.

### 5.3 Подписанные и конвертованные данные

Данные, пересылаемые между сторонами ИОК, защищаются с помощью алгоритмов ЭЦП и транспорта ключа. В этих алгоритмах используются открытые ключи из сертификатов сторон и соответствующие личные ключи.

Алгоритмы ЭЦП применяются для контроля целостности и подлинности данных. Данные подписываются на личном ключе отправителя. Корректность подписанных данных (т. е. данных с подписью) проверяется на открытом ключе из сертификата отправителя. Примером подписанных данных является сам сертификат.

Алгоритмы транспорта ключа применяются для обеспечения конфиденциальности данных. Сначала данные защищаются на случайном секретном ключе, а затем этот ключ защищается на открытом ключе получателя. Защищенный секретный ключ, называемый токеном ключа, присоединяется к защищенным данным, в результате чего образуются конвертованные данные. Получатель конвертованных данных сначала снимает защиту с

токена ключа на своем личном ключе, а затем использует полученный секретный ключ для снятия защиты с данных.

Универсальные форматы подписанных и конвертованных данных определены в СТБ 34.101.23. Форматы описываются типами `SignedData` и `EnvelopedData` АСН.1. Универсальность в том числе означает возможность инкапсуляции: подписанные данные могут быть конвертованы и наоборот.

Форматы `SignedData` и `EnvelopedData` профилируются в подразделах 8.6, 8.7. Речь идет о профилировании в рамках ИОК. За пределами ИОК форматы `SignedData` и `EnvelopedData` могут конкретизироваться по-другому.

### 5.4 Криптографические токены

Криптографические операции с личными ключами, используемые для создания подписанных данных и разбора конвертованных, реализуются в специализированных СКЗИ, называемых КТ.

КТ могут быть программными или аппаратными. Программный КТ представляет собой файл-контейнер с защищенным личным ключом и сопутствующими программами, которые реализуют криптографическую логику работы с ключом. Для защиты контейнера могут использоваться секретные данные, размещенные в других ключевых контейнерах КТ.

Программы КТ снимают защиту с контейнеров, извлекают личный ключ и работают с ним как с обычным объектом. Это может быть небезопасно, если программы выполняются в агрессивной среде, например, на общедоступном персональном компьютере.

Гарантии безопасности повышаются при переходе к аппаратному КТ. Здесь операции с личным ключом выполняются в пределах аппаратно защищенной криптографической границы токена. Личный ключ не покидает пределов границы, его секретность сохраняется даже при эксплуатации КТ в агрессивной среде.

Переход к аппаратному КТ не защищает от угрозы подмены данных, которые обрабатываются на личном ключе. От угрозы можно защититься, если организовать взаимодействие с токеном в терминальном режиме. В этом режиме КТ получает данные от терминала — стороны ИОК, предварительно прошедшей аутентификацию перед токеном. Данные передаются по защищенному соединению, и их невозможно раскрыть или подменить даже при полном контроле среды эксплуатации токена (но не терминала).

В качестве терминала может выступать локальное устройство или удаленный сервер. Кроме аутентификации перед КТ, терминал может проводить встречную аутентификацию КТ.

Стандартизация работы с КТ является еще одной задачей, которую решает настоящий стандарт. В разделе 11 определяется формат контейнера программного токена, в разделе 12 — программный интерфейс взаимодействия с аппаратным токеном.

В СТБ 34.101.79 определяются аппаратные КТ с расширенным функционалом. Эти токены дополнительно реализуют криптографические протоколы аутентификации, хранят идентификационные данные владельца и передают их доверенным сторонам в за-

щищенном виде. В СТБ 34.101.79 определяется низкоуровневый (командный) интерфейс работы с аппаратным КТ.

# 6 Криптографическая поддержка

## 6.1 Параметры эллиптической кривой

В криптографических алгоритмах ЭЦП и транспорта ключа должны использоваться стандартные параметры ЭК, установленные в СТБ 34.101.45 (приложение Б). Имеется три набора параметров для каждого из трех уровней стойкости $l$. В СТБ 34.101.45 (приложение Д) стандартным параметрам ЭК назначены идентификаторы `bign-curve256v1` ($l = 128$), `bign-curve384v1` ($l = 192$) и `bign-curve512v1` ($l = 256$). Уровень $l$, кроме параметров ЭК, определяет также длины личного и открытого ключей криптографических алгоритмов.

Параметры ЭК описываются типом `DomainParameters`, определенным в СТБ 34.101.45 (приложение Д). Для описания параметров должен быть выбран компонент `named` типа `DomainParameters`, и этот компонент должен содержать один из трех указанных выше идентификаторов.

## 6.2 Алгоритмы хэширования

Для хэширования подписываемых данных должен использоваться один из алгоритмов `belt-hash`, `bash384` или `bash512`. Алгоритм `belt-hash` установлен в СТБ 34.101.31, алгоритмы `bash384` и `bash512` — в СТБ 34.101.77. Идентификаторы алгоритмов определены в тех же стандартах. Алгоритмы возвращают соответственно 256-, 384- и 512-битовые хэш-значения.

## 6.3 Алгоритмы шифрования

Для шифрования конвертуемых данных должны использоваться либо алгоритмы `belt-cfb`, либо алгоритмы `belt-ctr`. Алгоритмы и их идентификаторы установлены в СТБ 34.101.31. Алгоритмы `belt-cfb` реализуют шифрование в режиме гаммирования с обратной связью, алгоритмы `belt-ctr` — в режиме счетчика.

В алгоритмах шифрования должны использоваться 256-битовые ключи.

## 6.4 Алгоритмы электронной цифровой подписи

Для проверки и выработки ЭЦП должны использоваться алгоритмы, установленные в СТБ 34.101.45. В алгоритмах должны использоваться стандартные параметры ЭК (см. 6.1), определяемые выбранным уровнем стойкости $l$.

Уровень стойкости, кроме параметров ЭК и длин ключей, определяет также вспомогательный алгоритм хэширования, уточняющий алгоритмы ЭЦП.

На уровне $l = 128$ вместе с параметрами `bign-curve256v1` должен использоваться алгоритм `belt-hash`. Уточненным алгоритмам ЭЦП назначается идентификатор `bign-with-hbelt`, определенный в СТБ 34.101.45 (приложение Д).

На уровне $l = 192$ вместе с параметрами `bign-curve384v1` должен использоваться алгоритм `bash384`. Уточненным алгоритмам ЭЦП назначается идентификатор `bign-with-bash384`, определенный в СТБ 34.101.77 (приложение Б).

На уровне $l = 256$ вместе с параметрами `bign-curve512v1` должен использоваться алгоритм `bash512`. Уточненным алгоритмам ЭЦП назначается идентификатор `bign-with-bash512`, определенный в СТБ 34.101.77 (приложение Б).

Формат ЭЦП определяется в СТБ 34.101.45 (приложение Д).

### 6.5 Алгоритмы транспорта ключа

Для транспорта ключа должны использоваться алгоритмы `bign-keytransport`. Алгоритмы и их параметры установлены в СТБ 34.101.45. В алгоритмах должны использоваться стандартные параметры ЭК (см. 6.1).

Формат защищенного транспортируемого ключа (токена ключа) определяется в СТБ 34.101.45 (приложение Д).

### 6.6 Описание алгоритмов

Для описания криптографических алгоритмов используется тип `AlgorithmIdentifier`, определенный в СТБ 34.101.19. Компонентами типа является идентификатор алгоритма (или пары связанных алгоритмов) и соответствующие параметры.

При описании алгоритмов `belt-hash`, `bash384`, `bash512`, `bign-with-hbelt`, `bign-with-bash384`, `bign-with-bash512` и `bign-keytransport` компонент параметров должен принимать значение `NULL`.

При описании алгоритмов `belt-cfb`, `belt-ctr` в компонент параметров записывается синхропосылка. Формат синхропосылки определяется в СТБ 34.101.31 (приложение Б).

### 6.7 Личный и открытый ключи

Для генерации личного и открытого ключей алгоритмов `bign-with-hbelt`, `bign-with-bash384`, `bign-with-bash512` и `bign-keytransport` должен использоваться алгоритм `bign-genkeypair`, определенный в СТБ 34.101.45. На вход `bign-genkeypair` подаются стандартные параметры ЭК одного из трех уровней стойкости.

Личный ключ уровня стойкости $l$ кодируется $l/4$ октетами, соответствующий открытый ключ — $l/2$ октетами. Правила кодирования определены в СТБ 34.101.45.

При генерации ключей субъекта сертификата следует учитывать, что уровень их стойкости не может быть выше уровня стойкости ключей эмитента.

Открытый ключ описывается типом `SubjectPublicKeyInfo`, определенным в СТБ 34.101.45 (приложение Д). Описание открытого ключа включает идентификатор параметров ЭК (см. 6.1). Если открытый и соответствующий личный ключи используются в алгоритмах ЭЦП, то идентификатор параметров ЭК, заданный в `SubjectPublicKeyInfo`, должен соответствовать идентификатору алгоритмов ЭЦП, заданному в `AlgorithmIdentifier`.

### 6.8 Билет и пароли

Для управления сертификатом предусмотрены два секрета: билет выпуска сертификата и пароль отзыва.

Билет выпуска представляет собой слово $P \in \{0,1\}^l$, где $l$ — уровень стойкости открытого ключа сертификата. Билет генерируется РЦ или УЦ с помошью физического генератора случайных чисел или алгоритма генерации псевдослучайных чисел, определенного в СТБ 34.101.47 или в другом ТНПА. Передается будущему субъекту сертификата и затем принимается от него в виде строки $\mathtt{hex}(P)$.

Пароль отзыва представляет собой текстовую строку, которая выбирается субъектом сертификата. Субъекту следует использовать высокоэнтропийные пароли: большой длины, с цифрами и буквами, без многократных повторов символов и т.д. Субъект ре-

гистрирует выбранный пароль в УЦ. УЦ может отказать в регистрации, если пароль не удовлетворяет определенным критериям качества.

Пароли используются также при работе с программным КТ для защиты контейнеров с частичными секретами (см. 11.1).

# 7 Стороны

## 7.1 Общие положения

Сторона ИОК характеризуется уникальными идентификационными данными. Сторона владеет одной или несколькими парами ключей (личным и открытым), каждой из которой соответствует сертификат, выпущенный в ИОК. Идентификационные данные стороны повторяются во всех ее сертификатах.

Несколько сертификатов может потребоваться стороне при взаимодействии с различными прикладными системами. В разных контекстах взаимодействия сторона может выступать в разных ролях, использовать ключи разных уровней стойкости (см. 6.1), управлять различными цифровыми ресурсами (см. 7.4). Кроме этого, логически единая сторона может состоять из нескольких компонентов, каждый из которых имеет собственную пару ключей и соответствующий сертификат, и при этом во всех сертификатах указываются единые идентификационные данные. Например, СШВ может представлять собой масштабируемый набор физических серверов, которые выпускают штампы времени от общего имени, используя собственные личные ключи.

В настоящем стандарте подчинение стороны $A$ стороне $B$ означает, что $A$ получила сертификат у $B$: $A$ является субъектом сертификата, $B$ — эмитентом. Речь идет о логическом подчинении в рамках ИОК. Подчинение не означает, что $A$ является подразделением $B$ или сотрудником $B$. Более того, подчиненная сторона $A$ может представлять ту же организацию, что и $B$. Создание в пределах одного ЮЛ нескольких сторон ИОК, возможно подчиненных друг другу, может использоваться для конкретизации сфер полномочий и обязанностей. Например, от РУЦ может быть логически отделен ПУЦ, отвечающий только за выпуск сертификатов для КТ.

## 7.2 Роли

Стороны ИОК делятся на УЦ, которые могут выпускать сертификаты открытых ключей, и конечных участников, которые выпускать сертификаты не могут.

В ИОК поддерживается следующая иерархия: имеется единственный КУЦ (точка доверия), единственный РУЦ и произвольное число ПУЦ. РУЦ подчиняется КУЦ, ПУЦ подчиняется РУЦ. Сертификаты конечным участникам выдает РУЦ либо ПУЦ. В первом случае участника связывает с КУЦ цепочка сертификатов длины 3, во втором — длины 4.

Определены следующие роли конечных участников ИОК:

1 ЦАС. Подчиняется РУЦ либо одному из ПУЦ. Выпускает атрибутные сертификаты для конечных участников, обслуживаемых некоторым (не обязательно своим) УЦ. Правила функционирования ЦАС определены в СТБ 34.101.67.

2 РЦ. Подчиняется РУЦ. Является посредником при взаимодействии между конечными участниками и УЦ: проводит первичную аутентификацию, регистрирует идентификационные данные, организует подготовку запросов на получение сертификата, визирует запросы. Для аутентификации и сбора идентификационных данных может взаимодействовать с КТ пользователей, выступая в качестве терминала. РЦ обслуживает один или

несколько УЦ. Может дополнительно обслуживать СИ, регистрируя данные или устройства для аутентификации (например, пароль или персональный КТ).

3 OCSP-сервер. Подчиняется РУЦ, либо одному из ПУЦ. По запросам сторон выпускает справки (OCSP-ответы) о текущем статусе сертификатов, выданных своим УЦ. В роли OCSP-сервера может выступать сам УЦ. Правила функционирования OCSP-сервера определены в СТБ 34.101.26. Форматы запроса и ответа профилируется в 8.8.

4 СШВ. Подчиняется РУЦ. По запросам сторон выпускает штампы времени, которые демонстрируют существование определенных данных к определенному моменту времени. Правила функционирования СШВ определены в СТБ 34.101.82. Формат штампов времени профилируется в 8.9.

5 СЗД. Подчиняется РУЦ. По запросам сторон выпускает справки (аттестаты заверения) о действительности ЭД. СЗД может проверять действительность ЭД в одной инфраструктуре и выпускать аттестат заверения в другой. Для проверки аттестата не нужно иметь доступ к первой инфраструктуре и, таким образом, СЗД позволяет организовать взаимодействие между сторонами инфраструктур, выступая в роли посредника между ними. Правила функционирования СЗД определены в СТБ 34.101.81. Используется только один из четырех сервисов СЗД — сервис vsd (проверка действительности ЭД). Формат аттестатов заверения профилируется в 8.10.

6 СИ. Подчиняется РУЦ, либо одному из ПУЦ. Проводит аутентификацию пользователей и, в случае успеха, выдает билеты — подписанные или секретные данные, которые открывают доступ пользователям к ресурсам прикладных систем или прикладным системам к ресурсам пользователей. Для организации аутентификации может взаимодействовать с КТ пользователя, выступая в качестве терминала.

7 TLS-сервер. Подчиняется РУЦ. Является частью прикладной системы, организует ее взаимодействие с другими сторонами по защищенным TLS-соединениям. Правила функционирования TLS-сервера описаны в СТБ 34.101.65.

8 ФЛ. Получает сертификат в РУЦ, либо в одном из ПУЦ. Может быть как резидентом, так и нерезидентом Республики Беларусь.

9 ЮП. Получает сертификат в РУЦ, либо в одном из ПУЦ. Представляет ЮЛ, несет ответственность за выполнение определенных процессов: технологических, юридических, организационных, финансовых и пр. Может быть оператором ПУД, подотчетного ЮЛ. ЮЛ-оператор должен получать сертификаты напрямую в РУЦ.

10 КА. Получает сертификат в РУЦ, либо в одном из ПУЦ. Может быть агентом ПУД, участвуя, например, в выпуске сертификатов.

УЦ может быть подчинено не более одного OCSP-сервера. СШВ и СЗД единственны (они подчинены РУЦ).

Операторы и агенты определенного УЦ должны получать сертификаты в этом УЦ.

Стороны ИОК представлены на рисунке 1. Роли, помеченные на рисунке звездочкой, могут быть представлены несколькими сторонами. Двойная линия означает однозначное соответствие между сторонами. Буква T означает, что сторона может выступать в качестве терминала. Скругленными прямоугольниками обозначены УЦ, наклоненными — службы.

Ролям назначаются идентификаторы АСН.1, определенные в приложении А. Идентификаторы имеют вид `{bpki-role n}`, где `bpki-role` — префикс, определенный в приложении, `n` — числовой код, заданный в таблице 1.

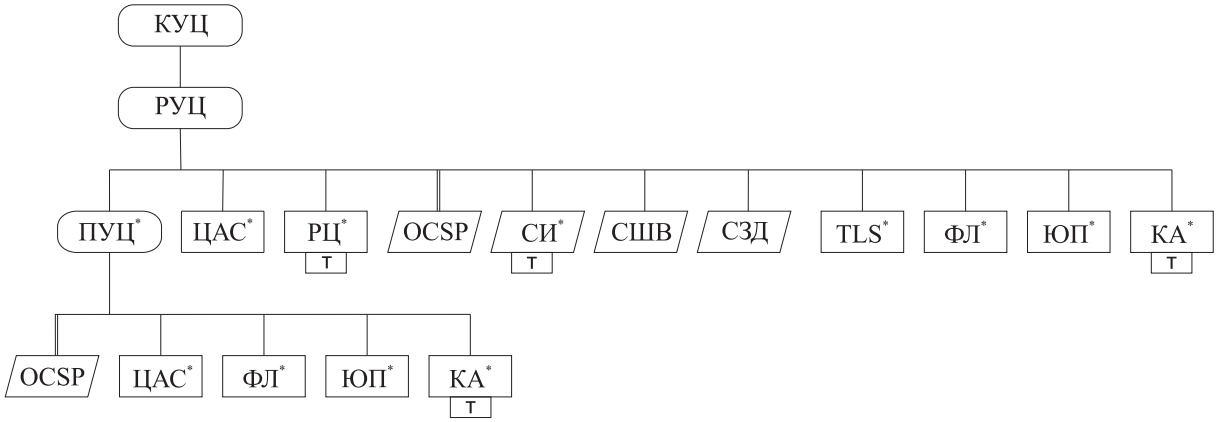


**Рисунок 1 — Стороны ИОК**

**Таблица 1 — Роли сторон**

| Роль | Код | Роль | Код |
|---|---|---|---|
| **КУЦ** | 0 | **СЗД** | 32 |
| **РУЦ** | 1 | **СИ** | 33 |
| **ПУЦ** | 2 | TLS-сервер | 50 |
| **ЦАС** | 10 | ФЛ-резидент | 60 |
| **РЦ** | 20 | ФЛ-нерезидент | 61 |
| **OCSP-сервер** | 30 | ЮП | 62 |
| **СШВ** | 31 | КА | 70 |

В таблице полужирным шрифтом выделены роли ПУД. Этим ролям зарезервирован диапазон кодов от 0 до 49.

Операторам ПУД назначается два идентификатора роли: идентификатор ЮП (основной) и идентификатор ПУД. Агентам ПУД также назначается два идентификатора: идентификатор КА (основной) и идентификатор ПУД. Дополнительный идентификатор ПУД имеет техническое значение: оператор сохраняет принадлежность роли ЮП, агент — роли КА.

Идентификаторы ролей конечных участников указываются в расширении `CertificatePolicies` их сертификатов (см. 8.1.5).

### 7.3 Идентификационные данные

#### 7.3.1 Идентификационные атрибуты

Идентификационные данные стороны представляют собой совокупность атрибутов: фамилия, имя и отчество, страна, место работы и др. Атрибут описывается типом `AttributeTypeAndValue`, определенным в СТБ 34.101.19, и представляет собой пару «идентификатор — значение». Идентификатор атрибута определяет его семантику, значение описывается строкой АСН.1 определенного типа с определенными ограничениями на длину.

Атрибуты укладываются в контейнер типа `Name`. Тип также определен в СТБ 34.101.19. Перечень атрибутов контейнера определяется ролью идентифицируемой стороны. В контейнере не должно быть нескольких однотипных атрибутов.

Тип `Name` имеют компоненты `subject` и `issuer` сертификата. Первый компонент описывает идентификационные данные субъекта, второй — эмитента.

Компонент `subject` типа `Name` включается также в запрос на получение сертификата. Указанные в компоненте идентификационные данные заверяются РЦ в процессе выпуска сертификата. Субъект не может изменить данные при самостоятельном (без участия РЦ) обновлении сертификата.

Перечень допустимых идентификационных атрибутов задан в таблице 2. Перечень составлен в соответствии с СТБ 34.101.19 и [5]. В сертификатах, издаваемых ПУЦ, могут указываться дополнительные идентификационные атрибуты.

**Таблица 2 — Идентификационные атрибуты**

| Атрибут | Идентификатор | Тип значения |
|---|---|---|
| commonName | {2 5 4 3} | UTF8String(SIZE (1..64)) |
| surname | {2 5 4 4} | UTF8String(SIZE (1..128)) |
| name | {2 5 4 41} | UTF8String(SIZE (1..1024)) |
| givenName | {2 5 4 42} | UTF8String(SIZE (1..128)) |
| serialNumber | {2 5 4 5} | PrintableString(SIZE (1..64)) |
| countryName | {2 5 4 6} | PrintableString(SIZE (2)) |
| localityName | {2 5 4 7} | UTF8String(SIZE (1..128)) |
| stateOrProvinceName | {2 5 4 8} | UTF8String(SIZE (1..128)) |
| organizationName | {2 5 4 10} | UTF8String(SIZE (1..64)) |
| organizationalUnitName | {2 5 4 11} | UTF8String(SIZE (1..64)) |
| title | {2 5 4 12} | UTF8String(SIZE (1..64)) |
| organizationIdentifier | {2 5 4 97} | UTF8String(SIZE (1..64)) |

Каждый из атрибутов представляет собой строку АСН.1 определенного типа. В строке должны отсутствовать незначащие пробелы, т. е. строка не должна начинаться с пробела, не должна заканчиваться пробелом и не должна содержать двух и более пробелов подряд.

В таблице 3 определяются атрибуты, которые должны быть включены в идентификационные данные сторон различных ролей. Пропуск в таблице означает отсутствие атрибута, плюс — обязательное включение, ± — включение при наличии.

**Таблица 3 — Идентификационные атрибуты ролей**

| Атрибут | ПУД | | TLS-сервер | ФЛ | | ЮП | КА |
|---|---|---|---|---|---|---|---|
| | КУЦ | другие | | резидент | нерезидент | | |
| commonName | + | + | + | + | + | + | + |
| surname | | | | + | + | + | |
| name* | | + | + | | | + | + |
| givenName | | | | + | + | + | |
| serialNumber | | | | + | + | + | + |
| countryName* | + | + | + | + | + | + | + |
| localityName* | | + | + | | | + | + |
| stateOrProvinceName* | | ± | ± | | | ± | ± |
| organizationName* | | + | + | | | + | + |
| organizationalUnitName* | | ± | ± | | | ± | ± |
| title | | | | | | + | |
| organizationIdentifier* | | + | + | | | + | + |

Атрибуты должны быть заданы в контейнере `Name` в том же порядке, в котором они представлены в таблице 3.

В первой колонке таблицы 3 звездочкой помечены идентификационные атрибуты операторов и агентов ПУД, которые должны повторять идентификационные атрибуты самого ПУД. Для остальных атрибутов операторов и агентов действуют правила ролей ЮП и КА соответственно.

### 7.3.2 Атрибут `commonName`

В атрибуте `commonName` задается общее (универсальное) имя стороны. В общем имени должны использоваться только графические символы базовой таблицы КОИ-7, определенной в ГОСТ 27463: латинские буквы, знаки препинания и базовые специальные знаки.

Общее имя следует выбирать так, чтобы оно кратко и при этом максимально однозначно характеризовало сторону. При выборе имени должны учитываться следующие ограничения:

1 Общее имя КУЦ полагается равным '`BY Root CA`'.

2 Общее имя РУЦ полагается равным '`BY Republican CA`'.

3 Общее имя СШВ полагается равным '`BY Republican TSA`'.

4 Общее имя СЗД полагается равным '`BY Republican DVCS`'.

5 Общее имя TLS-сервера содержит единичное или подстановочное (со звездочкой) DNS-имя. Примеры имен: '`www.example.org`', '`example.org`', '`*.example.org`'. Общее имя должно дублироваться в расширении `SubjectAltName` сертификата. В этом расширении могут быть указаны и другие DNS-имена.

6 Общее имя ФЛ или ЮП — это его имя и фамилия на английском языке в соответствии с удостоверением. Имя и фамилия записываются в верхнем регистре, разделяются пробелом. Например, '`VICTOR MITSKEVICH`'.

### 7.3.3 Атрибут `surname`

В атрибуте `surname` задается фамилия ФЛ или ЮП. Фамилия задается в соответствии с удостоверением лица.

Для ФЛ-резидентов и ЮП `surname` содержит белорусскую и русскую формы фамилии, записанные прописными буквами и разделенные наклонной чертой (как в паспорте). Например, '`МІЦКЕВІЧ/МИЦКЕВИЧ`'.

Примечание — Коды белорусских и русских символов определяются в соответствии с [9]. Код символа — это два октета, которые обычно записываются в шестнадцатеричной форме с префиксом `U-`. Например, `U-0406` — код белорусского символа І. Для сравнения, идентичный по начертанию латинский символ I задается другим кодом: `U-0049`. В настоящем стандарте белорусские и русские символы всегда размещаются в строках типа `UTF8String`. При этом символы дополнительно кодируются по правилам UTF-8, также определенным в [9]. Кодирование UTF-8 организовано так, что белорусские и русские символы снова представляются двумя октетами, а латинские символы — только одним.

### 7.3.4 Атрибут `name`

В атрибуте `name` задается полное название организации в соответствии с ее удостоверением. Например, '`Открытое акционерное общество "Вектор"`'.

Здесь и далее речь идет об организации, которая владеет ПУД или TLS-сервером, об организации, которая эксплуатирует КА, или об организации, которую представляет ЮП.

### 7.3.5 Атрибут givenName

В атрибуте givenName задается личное имя ФЛ или ЮП. Личное имя уточняет идентификацию лица на основе его фамилии. Имя задается в соответствии с удостоверением лица.

Для ФЛ-резидентов и ЮП givenName содержит белорусскую и русскую формы имени и, если имеется, отчества. Имя и отчество разделяются пробелом, записываются прописными буквами. Формы разделяются символом '/' (графический код байта $47 = 2F_{16}$ согласно базовой таблице КОИ-7). Например, 'ВІКТАР АНТОНАВІЧ/ВИКТОР АНТОНОВИЧ'.

### 7.3.6 Атрибут serialNumber

В атрибуте serialNumber задается либо идентификационный номер ФЛ или ЮП, либо серийный номер КА.

Строка, описывающая идентификационный номер, содержит (слева направо):

1 Три символа типа номера: 'PAS' — номер паспорта, 'PNO' — личный номер или 'IDC' — номер персонального аппаратного КТ (ID-карты). Идентификационный номер последнего типа должен использоваться только в сертификатах ФЛ и только тогда, когда соответствующий личный ключ размещается на персональном КТ.

2 Два символа кода страны, в которой зарегистрирован номер (см. 7.3.7).

3 Символ '-' (графический код байта $45 = 2D_{16}$).

4 Собственно символы номера.

Примеры: 'PASBY-MP0112358', 'PNOBY-786545091A4PB5', 'IDCBY-590082394654'.

Серийный номер КА следует задавать так, чтобы он однозначно характеризовал оборудование КА. Например, в качестве серийного номера может выступать MAC-адрес сетевого устройства или IMEI-номер мобильного телефона.

### 7.3.7 Атрибут countryName

В атрибуте countryName задается двухбуквенный код страны в соответствии с [6]. Для ФЛ-нерезидента это код страны, гражданином которой он является. Во всех остальных случаях код полагается равным 'BY'.

### 7.3.8 Атрибуты адреса

В атрибутах localityName и stateOrProvinceName задается информация из юридического адреса организации, которая владеет ПУД или TLS-сервером, или организации, которую представляет ЮП. Адрес задается в соответствии с удостоверением организации.

В атрибуте localityName указывается населенный пункт: город ('г.'), городской поселок ('г.п.'), деревня ('д.') и др. Примеры: 'г. Каменец', 'д. Каменюки'.

В атрибуте stateOrProvinceName указываются названия области и района. Атрибут не включается в идентификационные данные, если в localityName указан областной центр. В атрибуте опускается название района, если в localityName указан районный центр. Примеры: 'Брестская обл., Каменецкий р-н' (для 'д. Каменюки'), 'Брестская обл.' (для 'г. Каменец').

### 7.3.9 Атрибут organizationName

В атрибуте organizationName задается сокращенное название организации, которая владеет ПУД или TLS-сервером, или организации, которую представляет ЮП. Сокращенное название задается в соответствии с удостоверением организации. Например, 'ОАО "Вектор"'.

### 7.3.10 Атрибут `organizationalUnitName`

В атрибуте `organizationalUnitName` может задаваться название подразделения организации, которая отвечает за управление ПУД или TLS-сервером, или подразделения, которое представляет ЮП. Название подразделения задается в соответствии с удостоверением организации. Например, '`отдел цифровых технологий`'.

### 7.3.11 Атрибут `title`

В атрибуте `title` задается должность ЮП в организации, которую он представляет. Должность задается в соответствии с удостоверением ЮП. Например, '`начальник отдела`'.

### 7.3.12 Атрибут `organizationIdentifier`

В атрибуте `organizationIdentifier` задается идентификатор организации, которая владеет ПУД или TLS-сервером, или организации, которую представляет ЮП.

Строка, описывающая идентификатор, содержит (слева направо):

1 Три символа типа идентификатора. Разрешается использовать только код '`TAX`' — учетный номер плательщика.

2 '`BY`' — код страны, в которой зарегистрирован идентификатор.

3 Символ '`-`'.

4 Собственно символы идентификатора.

Пример: '`TAXBY-235831459`'.

## 7.4 Дополнительные идентификационные данные

В расширении `SubjectAltName` сертификата могут быть указаны дополнительные идентификационные данные. Эти данные представляют собой совокупность атрибутов, описывающих цифровые ресурсы стороны. Атрибуты задаются в компонентах вложенного в `SubjectAltName` типа `GeneralName`. Тип определен в СТБ 34.101.19. Перечень допустимых атрибутов задается таблицей 4.

**Таблица 4 — Дополнительные идентификационные атрибуты**

| Атрибут | Семантика (примеры) | Компонент `GeneralName` |
|---|---|---|
| `email` | адрес электронной почты (`alice@example.org`) | `rfc822Name` |
| `DNS` | DNS-имя (`www.example.org`, `*.example.org`) | `dNSName` |
| `URI` | URI (`http://example.org/responder`) | `uniformResourceIdentifier` |
| `IP` | IP-адрес (93.184.216.34) | `iPAddress` |

В расширении `SubjectAltName` должен присутствовать хотя бы один идентификационный атрибут. Может присутствовать несколько атрибутов одного типа. Общее число атрибутов не должно быть больше 100.

TLS-сервер должен указать в расширении `SubjectAltName` все свои DNS-имена. СШВ, СЗД, СИ и другие стороны могут указать в расширении URI своих сетевых узлов.

Расширение `SubjectAltName` включается в запрос на получение сертификата и переносится из запроса в сертификат. При заверении или обработке запроса оператору РЦ или УЦ следует проверить владение ресурсами, которые описываются атрибутами расширения. Способ проверки определяется вне рамок настоящего стандарта.

Дополнительные идентификационные атрибуты могут быть изменены при самостоятельном (без участия РЦ) обновлении сертификата.

# 8 Форматы данных

## 8.1 Расширения сертификата открытого ключа

### 8.1.1 Общие положения

Расширения сертификата описывают дополнительную информацию о субъекте, эмитенте или о самом сертификате. В зависимости от роли владельца сертификат может содержать те или иные наборы расширений.

В соответствии с СТБ 34.101.19 расширение может быть обязательным или необязательным, критическим или некритическим. Обязательное расширение — это расширение, которое должно присутствовать в сертификате. Критическое расширение — это обязательное расширение, которое должно быть корректным: при нарушении корректности сторона, проверяющая сертификат, должна завершить проверку с ошибкой.

Далее определяются допустимые расширения сертификатов, издаваемых КУЦ, РУЦ и ПУЦ. Если не оговорено противное, каждое из определяемых расширений является обязательным для каждой роли владельца сертификата.

Сертификаты, издаваемые ПУЦ, могут содержать расширения, дополнительные к перечисляемым ниже. Включение дополнительных расширений в сертификаты, издаваемые КУЦ и РУЦ, запрещено.

Названия расширений даются в соответствии с СТБ 34.101.19 (с прописной буквы). Исключение составляют расширения `ExtKeyUsage` и `AuthorityInfoAccess`, названия которых в СТБ 34.101.19 сопровождаются суффиксом `Syntax`.

### 8.1.2 Расширения `SubjectKeyIdentifier` и `AuthorityKeyIdentifier`

Расширения `SubjectKeyIdentifier` и `AuthorityKeyIdentifier` описывают хэш-значения открытых ключей субъекта и эмитента сертификата соответственно.

Расширения являются обязательными за одним исключением: `AuthorityKeyIdentifier` не должно включаться в самоподписанные сертификаты КУЦ. Расширения являются некритическими.

Хэш-значения должны вычисляться либо с помощью алгоритма `belt-hash`, определенного в СТБ 34.101.31, либо с помощью алгоритма SHA-1, определенного в [7]. В первом случае хэш-значение представляет собой строку из 32 октетов, во втором — строку из 20 октетов.

Примечание — Расширения `SubjectKeyIdentifier` и `AuthorityKeyIdentifier` облегчают построение цепочек сертификатов, не отвечая при этом за проверку цепочек. Поэтому в расширениях разрешается использовать алгоритм хэширования SHA-1, признанный на сегодняшний день криптографически нестойким. Разрешение на использование SHA-1 продиктовано необходимостью обеспечивать совместимость с действующими системами защиты информации. В тех случаях, когда совместимость не нужна, следует использовать `belt-hash`.

### 8.1.3 Расширение `KeyUsage`

Расширение `KeyUsage` описывает назначение открытого ключа. Описание представляет собой комбинацию флагов, определенных в СТБ 34.101.19.

Расширение является критическим.

В таблице 5 перечислены флаги `KeyUsage` для сторон различных ролей. Пропуск в таблице означает обязательное отсутствие флага, плюс — обязательное присутствие.

**Таблица 5 — Флаги** `KeyUsage`

| Сторона | digitalSignature | nonRepudiation | keyEncipherment | keyCertSign | cRLSign |
|---|---|---|---|---|---|
| КУЦ | | | | + | + |
| РУЦ | | | + | + | + |
| ПУЦ | | | + | + | + |
| ЦАС | + | | | | + |
| РЦ | + | + | + | | |
| OCSP-сервер | + | + | | | |
| СШВ | + | + | | | |
| СЗД | + | + | | | |
| СИ | + | + | + | | |
| TLS-сервер | + | | + | | |
| ФЛ | + | + | + | | |
| ЮП | + | + | + | | |
| КА | + | + | + | | |

### 8.1.4 Расширение `ExtKeyUsage`

Расширение `ExtKeyUsage` описывает область применения ключей сертификата. Описание представляет собой набор идентификаторов АСН.1.

Расширение `ExtKeyUsage` не должно включаться в сертификаты УЦ, ЦАС, РЦ и должно включаться в сертификаты остальных сторон. Расширение является критическим.

В сертификате OCSP-сервера расширение должно содержать идентификатор `id-kp-OCSPSigning`, определенный в СТБ 34.101.19.

В сертификате СЗД расширение должно содержать идентификатор `id-kp-dvcs`, определенный в СТБ 34.101.81.

В сертификате СШВ расширение должно содержать идентификатор `id-kp-timeStamping`, определенный в СТБ 34.101.19.

В сертификатах СИ и TLS-сервера расширение должно содержать идентификатор `id-kp-serverAuth`, определенный в СТБ 34.101.19.

В сертификатах ФЛ и ЮП расширение должно содержать идентификаторы `id-kp-clientAuth` и `id-kp-emailProtection`, определенные в СТБ 34.101.19.

В сертификате стороны, которая выступает в роли сервера (клиента) терминального режима, расширение `ExtKeyUsage` должно содержать идентификатор `bpki-eku-serverTM` (`bpki-eku-clientTM`). Идентификаторы определены в приложении А.

### 8.1.5 Расширение `CertificatePolicies`

Расширение `CertificatePolicies` описывают политику, в соответствии с которой был выпущен сертификат, и цели, в которых сертификат может использоваться.

Расширение `CertificatePolicies` не должно включаться в сертификаты КУЦ и должно включаться в сертификаты остальных сторон. Расширение является некритическим.

Описание политики состоит из пунктов, представленных идентификаторами ACH.1. В пунктах должны быть опущены опциональные классификаторы.

В сертификатах РУЦ и ПУЦ расширение должно содержать единственный пункт с идентификатором `anyPolicy`, определенным в СТБ 34.101.19.

В сертификате конечного участника расширение должно содержать пункты с идентификаторами его ролей. Идентификаторы определены в приложении А в соответствии с таблицей 1. Расширение может включать пункты нескольких ролей. Например, в сертификате оператора РЦ указываются две роли: ЮП и РЦ.

В сертификате конечного участника расширение `CertificatePolicies` может содержать дополнительные пункты, отличные от пунктов ролей. Например, пункт политики, в соответствии с которой проверялось владение TLS-сервером заявленным DNS-именем.

### 8.1.6 Расширение `BasicConstraints`

Расширение `BasicConstraints` дифференцирует сертификаты УЦ и конечных участников. Дополнительно расширение ограничивает длину цепочек сертификатов, подчиненных сертификату УЦ.

Расширение `BasicConstraints` является критическим.

В сертификатах УЦ флаг `cA` расширения должен быть установлен, в сертификатах конечных участников — сброшен.

В сертификатах КУЦ, РУЦ и конечных участников компонент `pathLenConstraint` должен отсутствовать, а в сертификате ПУЦ — принимать значение 0. Это значение означает запрет на выпуск ПУЦ сертификатов другим УЦ.

### 8.1.7 Расширение `SubjectAltName`

Расширение `SubjectAltName` содержит дополнительные идентификационные данные, описанные в 7.4.

Расширение является некритическим и необязательным.

### 8.1.8 Расширение `CRLDistributionPoints`

Расширение `CRLDistributionPoints` описывает расположение СОС, выпускаемых эмитентом сертификата.

Расширение `CRLDistributionPoints` не должно включаться в сертификаты КУЦ и должно включаться в сертификаты остальных сторон. Расширение является некритическим и обязательным.

Отдельные точки распространения списков отзыва описываются типом `DistributionPoint`. Опциональные компоненты `reasons` и `cRLIssuer` этого типа должны быть опущены, а в компоненте `distributionPoint` должен быть указан URI-адрес точки распространения (через выбор сначала варианта `fullName`, а затем варианта `uniformResourceIdentifier`).

### 8.1.9 Расширение `AuthorityInfoAccess`

Расширение `AuthorityInfoAccess` описывает информационные ресурсы, связанные с эмитентом сертификата. Расширение состоит из пунктов типа `AccessDescription`. Каждый такой пункт, в свою очередь, состоит из компонентов `accessMethod` (тип ресурса) и `accessLocation` (расположение ресурса).

Расширение не должно включаться в сертификаты КУЦ и РУЦ и должно включаться в сертификаты остальных сторон. Расширение является некритическим.

Первый пункт расширения описывает расположение сертификатов (одного или нескольких) эмитента. В компоненте `accessMethod` пункта должен быть установлен идентификатор `id-ad-caIssuers`, определенный в СТБ 34.101.19, а в компоненте `accessLocation` — URI-адрес сертификатов эмитента (через выбор варианта `uniformResourceIdentifier`).

Второй пункт расширения описывает расположение OCSP-сервера, к которому следует обратиться, чтобы получить информацию о статусе текущего сертификата. В компоненте `accessMethod` пункта должен быть установлен идентификатор `id-ad-ocsp`, определенный в СТБ 34.101.19, а в компоненте `accessLocation` — URI-адрес OCSP-сервера (через выбор варианта `uniformResourceIdentifier`).

## 8.2 Запрос на получение сертификата

### 8.2.1 Структура

Запрос на получение сертификата описывается типом `CertificationRequest`, который определен в СТБ 34.101.17. Компонент `subject` запроса должен быть составлен в соответствии с требованиями раздела 7.3.

В запрос могут быть включены атрибуты, которые описываются типом `Attribute`, также определенном в СТБ 34.101.17. Атрибут представляет собой пару «идентификатор типа (компонент `type`) — значение (компонент `value`)».

Разрешается использовать следующие атрибуты:

1 Атрибут `challengePassword`. Определен в [8]. Идентификатор типа равняется `{1 2 840 113549 1 9 7}`. Значение — строка типа `UTF8String(SIZE(1..255))`.

2 Атрибут `extensionRequest`. Определен в [8]. Идентификатор типа равняется `{1 2 840 113549 1 9 14}`. Значение — структура типа `Extensions`, определенного в СТБ 34.101.19. В компонентах `Extensions` указываются расширения сертификата, которые планируется перенести в сертификат.

3 Атрибут `certificateValidity`. Идентификатор типа равняется `bpki-at-certificateValidity` (определен в приложении А). Значение — структура типа `Validity`, определенного в СТБ 34.101.19. В компонентах `notBefore` и `notAfter` контейнера `Validity` указываются соответственно даты начала и окончания действия сертификата, рекомендуемые для переноса в сертификат. УЦ может принять рекомендации полностью, частично, или вообще проигнорировать.

В запросах к ПУЦ могут содержаться дополнительные атрибуты.

### 8.2.2 Атрибут `challengePassword`

Строка-значение атрибута `challengePassword` состоит из двух частей:

1 Билет выпуска сертификата. Используется в сценарии `Enroll3` процесса `Enroll` (см. 9.2), доказывая полномочия на выпуск.

2 Информационная строка, которую требуется передать УЦ. Например, реквизиты платежного документа об оплате услуги (процесса).

Первая часть атрибута представляет собой строку длины 32, 48 или 64 в алфавите $\{'0', '1', \ldots, 'F'\}$ (см. 6.8). Длина второй части не должна превосходить 128. Любая из частей может быть опущена. Порядок частей не контролируется. Части одного типа не должны повторяться.

Билету выпуска должен предшествовать префикс '/EPWD:', информационной строке — префикс '/INFO:'.

Пример атрибута: '/EPWD:01234...EF/INFO:SN112358'.

### 8.2.3 Атрибут extensionRequest

В атрибуте extensionRequest могут быть указаны следующие расширения сертификата:

1 ExtKeyUsage. Расширение указывается в тех случаях, когда будущий субъект планирует выступать в роли сервера / клиента терминального режима. Соответственно расширение может содержать только идентификаторы bpki-eku-serverTM / bpki-eku-clientTM (см. 8.1.4).

2 SubjectAltName. В расширении указываются дополнительные идентификационные атрибуты (см. 7.4). Расширение должны включать в свои запросы TLS-сервер и КА, могут включать другие стороны.

3 CertificatePolicies. В расширении указываются идентификаторы ролей будущего субъекта сертификата (см. 8.1.5). Расширение должны включать в свои запросы конечные участники и не должны включать РУЦ и ПУЦ.

В запросах к ПУЦ атрибут extensionRequest может содержать дополнительные расширения.

## 8.3 Сертификат открытого ключа

Формат сертификата открытого ключа описывается типом Certificate, который определен в СТБ 34.101.19.

Идентификатор алгоритмов ЭЦП, указываемый в компоненте signatureAlgorithm основного контейнера Certificate и дублируемый в компоненте signature вложенного контейнера TBSCertificate, должен быть выбран из перечня, заданного в 6.4. Этот идентификатор должен соответствовать открытому ключу эмитента сертификата (см. 6.7).

Вложенный контейнер TBSCertificate заполняется по правилам СТБ 34.101.19 со следующими уточнениями.

1 Серийный номер (компонент serialNumber) должен быть положительным целым числом, DER-код которого укладывается в 20 октетов. УЦ должен гарантировать уникальность серийных номеров всех выпускаемых сертификатов, даже тех, которые подписываются на разных личных ключах УЦ.

2 Идентификационные данные эмитента и субъекта (компоненты issuer и subject) заполняются в соответствии с 7.3.

3 Продолжительность действия сертификата, определяемая компонентом validity, не должна превышать значений, указанных в таблице 6. Максимальный срок действия сертификата в таблице определяется в зависимости от роли субъекта и уровня стойкости его ключей. При этом, как обычно, сертификаты операторов подчиняются правилам для ЮП, а сертификаты агентов — правилам для КА. Ограничения таблицы 6 не должны нарушаться при продлении сертификата с сохранением открытого ключа (см. 9.3).

**Таблица 6 — Сроки действия сертификатов**

| Роль | Уровень стойкости | Максимальный срок действия (лет) |
|---|---|---|
| КУЦ | $l = 128$ | 20 |
| | $l = 192$ | 30 |
| | $l = 256$ | 40 |
| РУЦ | $l = 128$ | 15 |
| | $l = 192$ | 20 |
| | $l = 256$ | 30 |
| ПУЦ, СШВ, СЗД, ЦАС, РЦ | $l = 128$ | 5 |
| | $l = 192$ | 8 |
| | $l = 256$ | 10 |
| OCSP, TLS, СИ, КА | $l = 128$ | 3 |
| | $l = 192$ | 4 |
| | $l = 256$ | 5 |
| ФЛ, ЮП | $l = 128$ | 2 |
| | $l = 192$ | 3 |
| | $l = 256$ | 4 |

4 Открытый ключ субъекта (компонент `subjectPublicKeyInfo`) должен описываться по правилам, заданным в 6.7.

5 Должен присутствовать опциональный компонент `extensions` и быть опущены остальные опциональные компоненты. Компонент `extensions` должен заполняться по правилам, заданным в 8.1.

### 8.4 Запрос на отзыв сертификата

Формат запроса на отзыв сертификата определяется следующим типом АСН.1:

```
BPKIRevokeReq ::= SEQUENCE {
  issuer          Name,
  serialNumber    INTEGER,
  revokePwd       UTF8String,
  reasonCode      CRLReason,
  invalidityDate  GeneralizedTime OPTIONAL,
  comment         UTF8String OPTIONAL }
```

Компонент `issuer` содержит идентификационные данные эмитента отзываемого сертификата.

Компонент `serialNumber` содержит серийный номер отзываемого сертификата.

Компонент `revokePwd` содержит пароль отзыва сертификата, который является действительным в настоящий момент. Этот пароль должен быть предварительно установлен с помощью процесса `Setpwd` (см. 9.6).

Компоненты `reasonCode` и `invalidityDate` содержат рекомендации УЦ по заполнению записи об отзыве сертификата (см. описание одноименных компонентов в 8.5). УЦ может учесть рекомендации или проигнорировать их.

Опциональный компонент `comment` содержит дополнительную информацию о причине отзыва.

### 8.5 Список отозванных сертификатов

СОС издает УЦ, ранее выпустивший отзываемые сертификаты. Формат СОС описывается типом `CertificateList`, который определен в СТБ 34.101.19.

Идентификатор алгоритмов ЭЦП, указываемый в компоненте `signatureAlgorithm` основного контейнера `CertificateList` и дублируемый в компоненте `signature` вложенного контейнера `TBSCertList`, должен быть выбран из перечня, заданного в 6.4. Этот идентификатор должен соответствовать открытому ключу издателя СОС (см. 6.7).

Вложенный контейнер `TBSCertList` заполняется по правилам СТБ 34.101.19 со следующими уточнениями.

1 Идентификационные данные издателя СОС, которые указываются в компоненте `issuer`, должны повторять данные в одноименном компоненте его сертификата.

2 В каждую запись об отозванном сертификате (компонент `revokedCertificates`) должно включаться расширение `reasonCode` с кодом причины отзыва сертификата. В запись может включаться расширение `invalidityDate`, которое описывает момент наступления события, повлекшего отзыв.

В расширении `reasonCode` могут использоваться следующие коды:

- `unspecified` — неопределенная причина;
- `keyCompromise` — компрометация личного ключа конечного участника (кроме ЦАС);
- `cACompromise` — компрометация личного ключа УЦ;
- `affiliationChanged` — смена идентификационных данных субъекта;
- `superseded` — смена сертификата (с помощью `Reenroll`, см. 9.3);
- `cessationOfOperation` — закрытие УЦ;
- `aACompromise` — компрометация личного ключа ЦАС.

3 В список расширений СОС (компонент `crlExtensions`) должны быть включены расширения `AuthorityKeyIdentifier` и `CRLNumber`. Первое расширение формируется по правилам, заданным в 8.1.2. Номер текущего СОС во втором расширении должен быть неотрицательным целым числом, DER-код которого укладывается в 20 октетов. Номера последовательных СОС должны монотонно возрастать.

### 8.6 Подписанные данные

Формат подписанных данных задается типом `SignedData`, который определен в СТБ 34.101.23.

Контейнер `SignedData` заполняется по правилам СТБ 34.101.23 со следующими уточнениями.

1 Версия синтаксиса (компонент `version`) должна равняться 1.

2 Список идентификаторов алгоритмов хэширования (компонент `digestAlgorithms`) должен содержать единственный элемент, и этот элемент должен быть выбран из перечня, заданного в 6.2.

3 Тип подписываемых данных (компонент `eContentType`, вложенный в `encapContentInfo`) должен принимать одно из следующих значений:

1) `id-ct-TSTInfo`, если подписывается ответ СШВ (см. 8.9.2);
2) `id-ct-DVCSResponseData`, если подписывается ответ СЗД (см. 8.10.2);
3) `bpki-ct-enroll1-req`, если в сценарии `Enroll1` процесса `Enroll` подписывается запрос на выпуск сертификата (см. 9.2.5);

4) `bpki-ct-enroll2-req`, если в сценарии `Enroll2` процесса `Enroll` подписывается запрос на выпуск сертификата (см. 9.2.5);
5) `bpki-ct-reenroll-req`, если в процессе `Reenroll` подписывается запрос на выпуск сертификата (см. 9.3);
6) `bpki-ct-spawn-req`, если в процессе `Spawn` подписывается запрос на выпуск сертификата (см. 9.4);
7) `bpki-ct-setpwd-req`, если в процессе `Setpwd` подписывается новый пароль (см. 9.6);
8) `bpki-ct-revoke-req`, если в процессе `Revoke` подписывается запрос на отзыв сертификата (см. 9.7);
9) `bpki-ct-resp`, если подписывается ответ УЦ (см. 8.11).

Первый идентификатор определен в СТБ 34.101.81, второй — в СТБ 34.101.82, остальные — в приложении А.

4 Опциональный компонент `certificates` должен присутствовать и должен содержать единственный сертификат — сертификат подписанта. Опциональный компонент `crls` должен быть опушен.

5 Список данных о подписантах (компонент `signerInfos`) должен содержать единственный контейнер типа `SignerInfo`, и этот контейнер должен быть заполнен следующим образом:

1) версия (компонент `version`) должна равняться 1;
2) идентификатор подписанта (`sid`) должен быть задан через тип `IssuerAndSerialNumber`. Компоненты `issuer` и `serialNumber` этого типа должны повторять одноименные компоненты сертификата подписанта;
3) идентификатор алгоритма хэширования (`digestAlgorithm`) должен совпадать с идентификатором, указанным в компоненте `digestAlgorithms` основного контейнера `SignedData`;
4) идентификатор алгоритмов ЭЦП (`signatureAlgorithm`) должен быть выбран из перечня, заданного в 6.4. Алгоритмы ЭЦП должны соответствовать алгоритму хэширования из `digestAlgorithm` и открытому ключу подписанта;
5) в список подписанных атрибутов (`signedAttrs`) должны быть включены атрибуты «Тип содержимого» (`ContentType`), «Хэш-значение» (`MessageDigest`) и может быть включен атрибут «Время подписания» (`SigningTime`). Все атрибуты определены в СТБ 34.101.23;
6) список неподписанных атрибутов (`unsignedAttrs`) должен быть пуст.

### 8.7 Конвертованные данные

Формат конвертованных данных задается типом `EnvelopedData`, который определен в СТБ 34.101.23.

Контейнер `EnvelopedData` заполняется по правилам СТБ 34.101.23 со следующими уточнениями.

1 Версия синтаксиса (компонент `version`) должна равняться 2.

2 Опциональные компоненты `originatorInfo` и `unprotectedAttrs` должны быть опущены.

3 Идентификатор алгоритмов шифрования (компонент `contentEncryptionAlgorithm`, вложенный в `encryptedContentInfo`) должен быть выбран из перечня, заданного в 6.3.

4 Тип конвертуемых данных (компонент `eContentType`, вложенный в `encryptedContentInfo`) должен принимать одно из следующих значений:

1) `id-signedData`, если конвертуются подписанные данные;
2) `id-data`, если конвертуются неструктурированные данные: запрос на получение сертификата, сертификат, запрос на отзыв сертификата.

5 Список сведений о получателях (компонент `recipientInfos`) должен содержать единственный контейнер `RecipientInfo`, и в этом контейнере должен быть выбран компонент `ktri` типа `KeyTransRecipientInfo`.

Компонент `ktri` должен формироваться следующим образом:

1) версия применяемого синтаксиса (компонент `version`) должна равняться 2;
2) идентификатор получателя (`rid`) должен задаваться через выбор варианта `issuerAndSerialNumber`;
3) в `keyEncryptionAlgortithm` должен быть установлен идентификатор алгоритмов `bign-keytransport`, описанных в 6.5.

## 8.8 Запрос и ответ OCSP

### 8.8.1 Формат запроса

Формат запроса OCSP задается типом `OCSPRequest`, который определен в СТБ 34.101.26. В самом контейнере `OCSPRequest` и во вложенных в него контейнерах должен быть опущены все опциональные компоненты и компоненты со значениями по умолчанию.

Основная информационная часть `OCSPRequest` — это список ссылок на сертификаты, статус которых необходимо проверить. Каждая ссылка описывается контейнером `Request`, который заполняется по правилам СТБ 34.101.26. Идентификатор алгоритма хэширования, указанный в компоненте `hashAlgorithm` контейнера `Request`, должен выбираться из перечня, заданного в 6.2.

### 8.8.2 Формат ответа

Формат ответа OCSP задается типом `OCSPResponse`, который определен в СТБ 34.101.26.

Контейнер `OCSPResponse` заполняется по правилам СТБ 34.101.26 со следующими уточнениями.

1 В компоненте `signatureAlgorithm` контейнера `BasicOCSPResponse`, вложенного в `OCSPResponse`, должен быть указан алгоритм ЭЦП из перечня, заданного в 6.4.

2 Компонент `certs` контейнера `BasicOCSPResponse`, должен быть либо опущен, либо в этом компоненте должен быть указан единственный сертификат — сертификат отправителя OCSP-ответа.

3 В контейнере `ResponseData`, вложенном в `BasicOCSPResponse`, должны быть опущены компонент `version` со значением по умолчанию и опциональный компонент `responseExtensions`.

4 В каждом из контейнеров `SingleResponse`, вложенных в `ResponseData`, должен быть опущен компонент `singleExtensions`.

## 8.9 Запрос и ответ службы штампов времени

### 8.9.1 Формат запроса

Формат запроса на получение штампа времени задается типом `TimeStampReq`, который определен в СТБ 34.101.82.

В TimeStampReq должны быть опущены опциональные компоненты reqPolicy, nonce и extensions. Если в ответе не требуется получить сертификат СШВ, то должен быть также опущен компонент certReq, который по умолчанию принимает значение FALSE. Если сертификат нужен, то компонент certReq должен включаться со значением TRUE.

В компоненте messageImprint контейнера TimeStampReq указывается хэш-значение данных. Хэш-значение сопровождается описанием алгоритма хэширования. СШВ должна проверять соответствие алгоритма и длины хэш-значения. СШВ должна поддерживать, по крайней мере, алгоритмы belt-hash, bash384 и bash512 (см. 6.2).

### 8.9.2 Формат ответа

Формат ответа СШВ задается типом TimeStampResp, который определен в СТБ 34.101.82.

Статус обработки запроса указывается в компоненте status. В случае успеха вложенные в status компоненты statusInfo и failInfo должны быть опущены, а вложенный компонент status должен принимать значение granted. Если ошибка произошла по причине загруженности сервера, то statusInfo и failInfo также должны быть опущены, а status должен принимать значение waiting. При других ошибках в status должно быть выбрано значение rejection, компонент statusInfo должен быть опущен, а в failInfo должен быть указан один из кодов ошибки, определенных в СТБ 34.101.82.

Штамп времени указывается в компоненте timeStampToken. Штамп, в свою очередь, содержит контейнер типа SignedData. Контейнер SignedData должен формироваться по правилам, заданным в 8.6. Дополнительное правило касается компонента certificates с сертификатом СШВ: этот компонент должен отсутствовать, если в запросе на получение штампа времени опущен флаг certReq.

Контейнер SignedData должен содержать ссылку на сертификат СШВ в виде подписанного атрибута SigningCertificate или SigningCertificateV2. Эти атрибуты определены в СТБ 34.101.80. В SigningCertificate должен быть опущен компонент policies, должен быть вложен только один контейнер ESSCertID (ссылка на сертификат СШВ) и в этом контейнере должен быть опущен компонент issuerSerial. Аналогично, в SigningCertificateV2 должен быть опущен компонент policies, в единственном вложенном контейнере ESSCertIDv2 должен быть опущен компонент issuerSerial.

В SignedData непосредственно подписывается контейнер типа TSTInfo. В подписываемом контейнере должны быть опущены все опциональные компоненты кроме, возможно, accuracy. В компоненте version должно быть установлено значение 1, а в компоненте policy — значение bpki-role-tsa, определенное в приложении А. В компонент messageImprint должно быть перенесено значение одноименного компонента запроса.

## 8.10 Запрос и ответ службы заверения данных

### 8.10.1 Формат запроса

СЗД должна поддерживать сервис проверки действительности ЭД (vsd) и не должна поддерживать другие сервисы.

Формат запроса СЗД задается типом DVCSRequest, который определен в СТБ 34.101.81. Запрос содержит общие данные запроса и заверяемый ЭД.

Общие данные запроса указываются в компоненте requestInformation типа DVCSRequestInformation. Тип описывает версию применяемого синтаксиса и запраши-

ваемый клиентом сервис. Должны быть выбраны версия 1 и сервис vsd. Все опциональные компоненты `DVCSRequestInformation` должны быть опущены.

Заверяемый ЭД кодируется строкой октетов и указывается в компоненте `message`, вложенном в компонент `data` типа `DVCSRequest`.

### 8.10.2 Формат ответа

Формат ответа СЗД задается контейнером типа `SignedData`. Контейнер должен формироваться по правилам, заданным в 8.6.

Непосредственно подписывается значение типа `DVCSResponse`. Это значение инкапсулируется в `SignedData` с идентификатором `id-ct-DVCSResponseData`.

СЗД должна выносить один из трех вердиктов: `granted` — документ признан действительным, `waiting` — проверка не завершена, `rejection` — документ признан недействительным. При первых двух вердиктах в `DVCSResponce` должен быть выбран компонент `dvCertInfo`, в случае третьего вердикта — компонент `dvErrorNote`.

В компоненте `dvCertInfo` должны быть опущены все вложенные компоненты, кроме `version`, `dvReqInfo`, `messageImprint`, `serialNumber`, `responseTime`. Вложенный компонент `dvStatus` должен присутствовать в случае вердикта `waiting` и должен быть опущен при вердикте `granted`.

В компоненте `dvErrorNote` должен быть опущен вложенный компонент `transactionIdentifier`.

Для компонентов `dvCertInfo` должны соблюдаться следующие правила:

1 Компонент `version` должен принимать значение 1.

2 В `dvReqInfo` должна быть перенесена общая часть запроса.

3 Алгоритм хэширования, указанный в `messageImprint`, должен соответствовать алгоритму ЭЦП, который используется для подписи ответа.

## 8.11 Ответ удостоверяющего центра

Формат ответа УЦ на запрос другой стороны определяется следующим типом ASN.1:

```
BPKIResp ::= SEQUENCE {
  statusInfo  PKIStatusInfo,
  requestId   OCTET STRING(SIZE(32)),
  nonce       OCTET STRING(SIZE(8)) OPTIONAL }
```

Компонент `statusInfo` содержит статус обработки запроса. Тип `PKIStatusInfo` определен в СТБ 34.101.82.

В случае успеха вложенные в `statusInfo` компоненты `statusString` и `failInfo` должны быть опущены, а вложенный компонент `status` должен принимать значение `granted`. Если ошибка произошла по причине загруженности УЦ или потому, что обработка запроса требует времени, то `statusString` и `failInfo` также должны быть опущены, а `status` должно принимать значение `wating`. При других ошибках в `status` должно быть выбрано значение `rejection`, в `failInfo` должен быть указан один из кодов ошибки, определенных в СТБ 34.101.82, а в `statusString` ошибка может быть дополнительно прокомментирована.

Компонент `requestId` содержит идентификатор запроса — его хэш-значение, вычисляемое с помощью алгоритма `belt-hash` (см. 6.2).

Опциональный компонент `nonce` содержит синхропосылку — случайную строку октетов, которая делает ответы УЦ неповторяющимися даже при одинаковых `requestId`. Компонент используется только в ответах на повторные запросы.

### 8.12 Повторный запрос

Если получен ответ УЦ со статусом `waiting`, то этот ответ можно уточнить, обратившись к УЦ повторно. Формат повторного запроса определяется следующим типом АСН.1:

```
BPKIRetrieveReq ::= SEQUENCE {
  requestId   OCTET STRING(SIZE(32)),
  nonce       OCTET STRING(SIZE(8))}
```

Компонент `requestId` содержит идентификатор первоначального запроса, компонент `nonce` — синхропосылку. Синхропосылка выбирается случайно отправителем повторного запроса.

При обработке повторного запроса `BPKIRetrieveReq` УЦ переносит его компоненты в свой ответ `BPKIResp`.

## 9 Процессы

### 9.1 Перечень процессов

Управление сертификатами конечных участников ИОК реализуется через следующие процессы.

1 `Enroll` — выпуск сертификата для стороны, которая располагает действительным удостоверением, но не обязательно действительным сертификатом.

2 `Reenroll` — обновление действительного сертификата.

3 `Spawn` — выпуск нового сертификата для стороны, которая располагает действительным сертификатом.

4 `Retrieve` — получение сертификата, который был запрошен в процессах `Enroll`, `Reenroll`, `Spawn` и выпуск которого задерживается.

5 `Setpwd` — установка (изменение) пароля отзыва сертификата.

6 `Revoke` — отзыв сертификата.

Каждый процесс представляет собой последовательность определенных процедур. При ошибке в любой из процедур процесс также завершается с ошибкой.

В каждом процессе обязательно участвует УЦ: РУЦ или ПУЦ. Процесс включает процедуры отправки запроса УЦ и получения соответствующего ответа. Сторона, отправляющая запрос, должна располагать сертификатом УЦ.

Форматы запросов и ответов схематически представлены в таблице 7. Используются имена типов данных и их компонентов, описанные в разделе 8. Нижний индекс С означает субъекта сертификата, нижний индекс опРЦ — оператора РЦ, нижний индекс агУЦ — агента УЦ. Нижний индекс у контейнера `SignedData` указывает на подписанта, у контейнера `EnvelopedData` — на получателя конвертованных данных, у других объектов — на их владельцев.

Перечисленные в таблице запросы и ответы транспортируются в виде пакетов HTTP. Правила транспорта определяются в 10.

**Таблица 7 — Схемы форматов запросов / ответов**

| Процесс `Enroll` | |
|---|---|
| Запрос | $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{SignedData}_{\text{РЦ} \mid \text{опРЦ}}(\texttt{CertificateRequest}_{\text{C}}))$ или $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{CertificateRequest}_{\text{C}}[\texttt{enrollPwd}])$ |
| Ответ | $\texttt{EnvelopedData}_{\text{РЦ} \mid \text{опРЦ} \mid \text{C}}(\texttt{Certificate}_{\text{C}})$ или $\texttt{SignedData}_{\text{агУЦ}}(\texttt{BPKIResp})$ |
| Процессы `Reenroll` и `Spawn` | |
| Запрос | $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{SignedData}_{\text{C}}(\texttt{CertificateRequest}_{\text{C}}))$ |
| Ответ | $\texttt{EnvelopedData}_{\text{C}}(\texttt{Certificate}_{\text{C}})$ или $\texttt{SignedData}_{\text{агУЦ}}(\texttt{BPKIResp})$ |
| Процесс `Retrieve` | |
| Запрос | `BPKIRetrieveReq` |
| Ответ | $\texttt{EnvelopedData}_{\text{РЦ} \mid \text{опРЦ} \mid \text{C}}(\texttt{Certificate}_{\text{C}})$ или $\texttt{SignedData}_{\text{агУЦ}}(\texttt{BPKIResp})$ |
| Процесс `Setpwd` | |
| Запрос | $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{SignedData}_{\text{C}}(\texttt{revokePwd}))$ |
| Ответ | $\texttt{SignedData}_{\text{агУЦ}}(\texttt{BPKIResp})$ |
| Процесс `Revoke` | |
| Запрос | $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{SignedData}_{\text{C}}(\texttt{BPKIRevoke}))$ или $\texttt{EnvelopedData}_{\text{УЦ}}(\texttt{BPKIRevoke})$ |
| Ответ | $\texttt{SignedData}_{\text{агУЦ}}(\texttt{BPKIResp})$ |

### 9.2 Процесс `Enroll`

#### 9.2.1 Сценарии

В процессе `Enroll` участвуют УЦ и сторона, запрашивающая сертификат (будущий его субъект). Дополнительно могут быть задействованы РЦ или его оператор, а также агент УЦ.

Процесс состоит из следующих процедур:

- генерация ключей;
- подготовка запроса на получение сертификата;
- аутентификация субъекта;
- заверение запроса;
- отправка запроса УЦ;
- обработка запроса;
- возврат ответа;
- обработка ответа.

Процесс может конфигурироваться: одни и те же процедуры могут выполняться разными сторонами, процедуры могут опускаться, может меняться последовательность процедур. При конфигурировании могут быть реализованы следующие сценарии.

`Enroll1`. Субъект самостоятельно генерирует ключи и готовит запрос. Оператор РЦ проводит аутентификацию, заверяет и отправляет запрос, обрабатывает ответ. Результат передается субъекту.

`Enroll2`. Субъект взаимодействует с РЦ с помощью аппаратного КТ. РЦ в качестве терминала проводит аутентификацию КТ, вызывает команду генерации ключей, извлекает идентификационные данные и открытый ключ, готовит запрос, заверяет и отправляет его УЦ. РЦ обрабатывает ответ УЦ и записывает результат обработки на КТ. Ключи генерируются внутри КТ субъекта, хотя генерацию инициирует РЦ.

`Enroll3`. Субъект предварительно проходит аутентификацию, регистрирует свои идентификационные данные и получает билет `enrollPwd`, который указывает в своем запросе. Субъект самостоятельно готовит и отправляет запрос, обрабатывает ответ.

Первый сценарий является стандартным способом выпуска первого сертификата субъекта. Его недостаток — невозможность реализации онлайн, без визита в подразделение РЦ. Избежать визита можно с помощью второго сценария, но при этом необходимо располагать КТ. Третий сценарий ориентирован на выпуск сертификатов КА. Сценарий может использоваться также для выпуска сертификатов ПУД и их операторов, в частности, операторов РЦ, задействованных в `Enroll1`.

### 9.2.2 Генерация ключей

Генерируются личный и открытый ключи СТБ 34.101.45. Правила генерации определены в 6.7.

В `Enroll1`, `Enroll3` ключи генерирует сам субъект (владелец ключей) с помощью программного или аппаратного КТ. Сгенерированный личный ключ сохраняется либо в ключевом контейнере, либо внутри аппаратного КТ. Открытый ключ может не сохраняться, при необходимости он вычисляется по личному ключу.

В `Enroll2` ключи генерируются внутри аппаратного КТ субъекта, но запрос на генерацию дает РЦ, взаимодействующий с КТ в терминальном режиме.

### 9.2.3 Подготовка запроса

После генерации пары ключей готовится запрос на получение сертификата. Формат запроса и правила его заполнения описаны в 8.2. В запрос включается сгенерированный открытый ключ. Запрос подписывается на соответствующем личном ключе.

В `Enroll1` субъект готовит запрос самостоятельно. Субъект переносит в запрос идентификационные данные из своих удостоверений. Перенос может быть выполнен с помощью оператора РЦ.

Субъекту рекомендуется включить в атрибут `extensionRequest` запроса расширение `subjectAltName` и указать в нем свой адрес электронной почты (см. 7.4).

Субъект может включить в запрос атрибут `challengePassword` (см. 8.2.2). Этот атрибут должен содержать информационную строку (префикс `'/INFO:'`), которая позволит УЦ проверить факт оплаты услуг доверия, другие факты.

Если в удостоверениях есть ограничения, которые сужают предполагаемый срок действия сертификата, то субъекту следует включить в запрос атрибут `certificateValidity`, который проинформирует УЦ о сужении.

В `Enroll2` запрос готовит РЦ, который взаимодействует с КТ субъекта в терминальном режиме. РЦ получает от КТ идентификационные данные субъекта и сгенерированный открытый ключ. Данные от КТ пересылаются по защищенному соединению после взаимной аутентификации КТ и РЦ. При формировании запроса РЦ может включить в него расширение `ExtKeyUsage` с флагом `bpki-eku-clientTM` (см. 8.2.3) и, таким образом, запросить выдачу терминального сертификата.

В атрибуте `SerialNumber` идентификационных данных субъекта должен быть указан номер ID-карты (префикс '`IDCBY-`'). Перенос этого номера в сертификат будет означать, что сертификат обслуживается личным ключом персонального КТ.

РЦ может включить в запрос атрибут `certificateValidity` и указать в нем планируемый срок действия сертификата. Этот срок не должен выходить за рамки действия персонального КТ.

В `Enroll3` субъект готовит запрос самостоятельно. Субъект повторяет в запросе идентификационные данные, зарегистрированные в ИОК при предварительной аутентификации (см. 9.2.4). Субъект обязательно указывает в запросе билет `enrollPwd`, полученный после аутентификации. Этот билет задается в атрибуте `challengePassword` запроса (см. 8.2.2) как билет выпуска (префикс '`/EPWD:`').

### 9.2.4 Аутентификация субъекта

Аутентификация состоит в проверке подлинности субъекта. Аутентифицируется либо непосредственно субъект, либо его аппаратный КТ.

В `Enroll1` аутентификацию субъекта проводит оператор РЦ. При аутентификации проверяются удостоверения субъекта.

Если в удостоверениях есть ограничения по срокам действия, то оператору РЦ следует проверить включение в запрос атрибута `certificateValidity` и его соответствие удостоверениям.

В `Enroll2` аутентификацию аппаратного КТ проводит РЦ, который выступает в качестве терминала. Субъектом при этом может быть только ФЛ-резидент, для которого КТ является удостоверением. Правила аутентификации определены в СТБ 34.101.79. Аутентификация КТ выполняется до генерации ключей.

В `Enroll3` проводится предварительная аутентификация субъекта с одновременной регистрацией его идентификационных данных и идентификаторов ролей (будут указаны в расширении `CertificatePolicies` сертификата). При успешной регистрации субъект получает билет `enrollPwd`. Тройки «идентификационные данные — идентификаторы ролей — пароль» передаются УЦ и хранятся в ожидании соответствующего запроса на получение сертификата. УЦ должен хранить тройки не менее 90 суток.

В `Enroll3` аутентификацию и регистрацию проводит оператор РЦ или оператор УЦ. Если субъектом является КА или ПУД, то аутентифицироваться вместо них может ЮП соответствующей организации. Способ доставки УЦ зарегистрированных данных в настоящем стандарте не детализируется.

### 9.2.5 Заверение запроса

Заверение состоит в подписи запроса. Заверяя запрос, подписывающая сторона подтверждает подлинность указанных в нем сведений. Подписанный запрос оформляется как контейнер `SignedData` с учетом правил, заданных в 8.6. Контейнер включает сертификат подписанта.

В Enroll1 запрос заверяет оператор РЦ. Оператор сверяет данные из удостоверений с данными в запросе. Оператору следует проверить владение ресурсами, описанными дополнительными идентификационными атрибутами email, DNS, URI, IP (см. 7.4).

В Enroll2 запрос заверяет РЦ. В сертификате РЦ расширение ExtKeyUsage (см. 8.1.4) должно содержать идентификатор bpki-eku-serverTM.

В Enroll3 запрос не заверяется.

### 9.2.6 Отправка запроса

Подписанный запрос отправляется УЦ. Перед отправкой запрос конвертуется на открытом ключе УЦ и отправляется в виде контейнера EnvelopedData. Формат контейнера описан в 8.7. Конвертование запроса обеспечивает конфиденциальность содержащихся в нем идентификационных данных субъекта.

Открытый ключ УЦ определяется по его сертификату. Перед отправкой должна проверяться действительность сертификата. Запрос не следует отправлять, если уровень стойкости указанного в нем открытого ключа выше уровня стойкости открытого ключа сертификата УЦ.

В Enroll1 отправку выполняет оператор РЦ, в Enroll2 — РЦ, в Enroll3 — сам субъект.

Перед отправкой запроса вычисляется его хэш-значение. В дальнейшем оно используется в качестве идентификатора запроса. Хэширование должно выполняться с помощью алгоритма belt-hash (см. 6.2). Идентификатор сохраняется вместе с ключевым контейнером или внутри аппаратного КТ. Идентификатор можно не вычислять, если сохранить сам запрос и хэшировать его при необходимости.

### 9.2.7 Обработка запроса

УЦ обрабатывает запрос по следующему алгоритму.

1 Вычислить идентификатор запроса, применив алгоритм хэширования belt-hash. Запомнить идентификатор.

2 Снять защиту с запроса как контейнера EnvelopedData и определить вложенные в контейнер данные.

3 Если в EnvelopedData вложен контейнер SignedData (сценарии Enroll1 или Enroll2), то выполнить следующие шаги:

1) найти в контейнере SignedData сертификат отправителя и проверить, что в его расширении CertificatePolicies установлена роль РЦ (см. таблицу 1);
2) связать открытый ключ из сертификата отправителя с идентификатором запроса;
3) проверить, что в контейнер SignedData вложены данные типа bpki-ct-enroll1-req (сценарий Enroll1) или bpki-ct-enroll2-req (сценарий Enroll2);
4) если в контейнер SignedData вложены данные типа bpki-ct-enroll2-req, то проверить, что расширение ExtKeyUsage сертификата отправителя содержит идентификатор bpki-eku-serverTM;
5) проверить действительность подписи контейнера SignedData (в том числе действительность сертификата отправителя);
6) извлечь из контейнера SignedData запрос на получение сертификата;
7) проверить подпись запроса на указанном в запросе открытом ключе;

8) проверить, что уровень стойкости открытого ключа запроса не выше уровня стойкости открытого ключа УЦ;
9) обработать информационную строку в атрибуте `challengePassword` запроса (при наличии).

4 Если в `EnvelopedData` вложен запрос на получение сертификата (сценарий `Enroll3`), то выполнить следующие шаги:

1) проверить подпись запроса на указанном в запросе открытом ключе;
2) проверить, что уровень стойкости открытого ключа запроса не выше уровня стойкости открытого ключа УЦ;
3) связать открытый ключ из запроса с идентификатором запроса;
4) сравнить указанные в запросе идентификационные данные с предварительно зарегистрированными;
5) сравнить указанные в запросе идентификаторы ролей (расширение `CertificatePoliciles` в атрибуте `extensionRequest`) с предварительно зарегистрированными;
6) проверить билет выпуска в атрибуте `challengePassword` запроса;
7) обработать информационную строку в атрибуте `challengePassword` запроса (при наличии).

5 Выпустить сертификат:

1) перенести в сертификат идентификационные данные субъекта из запроса;
2) указать в сертификате собственные идентификационные данные (как эмитента);
3) выбрать новый серийный номер сертификата;
4) установить начало действия сертификата – текущий момент времени;
5) окончание действия сертификата задать в соответствии с таблицей 6 и с учетом информационной строки, переданной через `challengePassword`;
6) если в запрос включен атрибут `certificateValidity`, то учесть рекомендованные в нем начало и окончание действия. Игнорировать рекомендуемое начало действия, если оно отстоит от текущего момента времени более чем на 30 суток. Игнорировать рекомендуемое окончание действия, если оно позже установленного на предыдущем шаге;
7) сформировать расширения сертификата. Использовать правила, изложенные в 8.1;
8) подписать сертификат.

При ошибке на любом из шагов алгоритма обработка запроса прекращается с возвратом соответствующего кода ошибки.

Кроме выходных сертификата или кода ошибки УЦ дополнительно фиксирует идентификатор запроса (шаг 1) и возможно связывает с ним открытый ключ (шаги 3.2, 4.3). Эти данные будут использоваться для подготовки ответа.

### 9.2.8 Возврат ответа

По результатам обработки запроса УЦ формирует ответ: сертификат или контейнер `BPKIResp`, формат которого описан в 8.11.

В `Enroll1`, `Enroll2` сертификат конвертуется на открытом ключе отправителя запроса. В `Enroll3` сертификат конвертуется на открытом ключе самого сертификата. В любом случае используется открытый ключ, который был связан с идентификатором запроса при обработке запроса.

Контейнер BPKIResp возвращается при ошибке во время обработки запроса. В компоненте requestId контейнера указывается идентификатор запроса, а в компоненте statusInfo описывается статус обработки запроса. Компонент nonce опускается. Контейнер BPKIResp подписывается, но не конвертуется. Подписанный ответ оформляется как контейнер SignedData.

Контейнер BPKIResp подписывается не самим УЦ, а его агентом. Взаимодействие УЦ и агента в настоящем стандарте не детализируется.

Контейнер BPKIResp возвращается также тогда, когда обработка запроса не была завершена к моменту ответа. В этом случае в контейнере указывается статус waiting.

### 9.2.9 Обработка ответа

Ответ УЦ получает та сторона, которая отправила запрос. Отправитель не может полагаться на то, что сертификат будет возвращен сразу (онлайн), даже если запрос корректен. Тем не менее, УЦ должен гарантировать возврат ответа на корректный запрос в течение 24 часов.

Ответ, который не является ни контейнером EnvelopedData, ни контейнером SignedData, признается некорректным.

Если ответ представляет собой контейнер EnvelopedData, то получатель снимает с него защиту на своем личном ключе, определяет вложенный сертификат и проверяет его действительность. В случае успеха сертификат передается субъекту (Enroll1), записывается на КТ субъекта (Enroll2), либо субъект сам получает свой сертификат (Enroll3).

В Enroll1 получатель (оператор РЦ) дополнительно проверяет наличие в сертификате адреса электронной почты (идентификационный атрибут email, см. 7.4). При наличии адреса сертификат должен быть отправлен по этому адресу. Сертификат может отправляться и в открытом, и в конвертованном виде. Конвертование выполняется на открытом ключе сертификата. В обозначениях таблицы 7 речь идет о контейнере $\texttt{EnvelopedData}_{\texttt{C}}(\texttt{Certificate}_{\texttt{C}})$. Способ отправки может предварительно согласовываться с субъектом при подготовке запроса на выпуск сертификата.

Если ответ представляет собой контейнер SignedData, то получатель проверяет тип его содержимого и подпись. Тип содержимого должен равняться bpki-ct-resp. Затем получатель определяет вложенный контейнер BPKIResp и анализирует его. Контейнер признается корректным, если указанный в нем идентификатор requestId совпадает с первоначальным идентификатором запроса. В конце концов получатель либо принимает ответ и обрабатывает указанный в нем статус обработки запроса, либо игнорирует ответ.

При проверке подписи контейнера SignedData дополнительно проверяется, что подписант является агентом целевого УЦ. Это значит, что должны выполняться следующие условия:

– сертификат подписанта выпущен целевым УЦ;

– идентификационные атрибуты субъекта сертификата, помеченные звездочкой в таблице 3, совпадают с атрибутами эмитента;

– в расширении CertificatePolicies сертификата установлены роли КА и УЦ.

Если в контейнере BPKIResp указан статус waiting, то сертификат все-таки может быть получен через повторное обращение к УЦ с помощью процесса Retrieve. В обращении должен использоваться идентификатор запроса. В Enroll1 идентификатор передается субъекту, в Enroll2 — записывается на КТ субъекта, в Enroll3 субъект сам получает идентификатор.

Возможные действия сторон при реализации сценариев выполнения процедур процесса `Enroll` приведены в таблице 8.

**Таблица 8 — Сценарии процесса `Enroll`**

| Процедура | Сценарий | | |
|---|---|---|---|
| | `Enroll1` | `Enroll2` | `Enroll3` |
| Генерация ключей | Генерирует субъект | Генерирует субъект под управлением РЦ | Генерирует субъект |
| Подготовка запроса | Самостоятельно субъект | РЦ при взаимодействии с КТ субъекта | Самостоятельно субъект |
| Аутентификация | Оператор РЦ | Терминал РЦ | Предварительная |
| Заверение запроса | Оператор РЦ | РЦ | – |
| Отправка запроса УЦ | Оператор РЦ | РЦ | Субъект |
| Обработка запроса | УЦ | УЦ | УЦ |
| Возврат ответа | УЦ. Ответ конвертуется на ключе отправителя | УЦ. Ответ конвертуется на ключе отправителя | УЦ. Ответ конвертуется на ключе сертификата |
| Обработка ответа | Сертификат передается субъекту | Сертификат записывается на КТ субъекта | Субъект сам получает сертификат |

### 9.3 Процесс `Reenroll`

Процесс `Reenroll` выполняют субъект сертификата и УЦ — его эмитент. С помощью `Reenroll` субъект продлевает срок действия сертификата или изменяет в нем дополнительные идентификационные атрибуты (см. 7.4). При успешном завершении `Reenroll` субъект получает новый сертификат, а его действующий сертификат отзывается.

В новый сертификат может быть перенесен открытый ключ действующего. При переносе сохраняется личный ключ, и субъект избавляется от необходимости менять ключевой контейнер или переписывать критическую память аппаратного КТ.

Поскольку `Reenroll` проходит без участия РЦ или его операторов, УЦ необходимо разработать и реализовать политику продления идентификационных данных на новый период действия сертификата. Частью политики могут быть обращения с запросами к специализированным информационным системам, например, регистру населения. Или УЦ может просто блокировать определенные запросы, например, запросы от ЮЛ, идентификационные данные которых более волатильны, чем у ФЛ.

Примечание — Обязательным элементом политики продления идентификационных данных является блокировка запросов относительно сертификатов, выпущенных с помощью `Enroll2`. Такие сертификаты выпускаются для персональных КТ с помощью РЦ в роли терминала. Новые сертификаты следует выпускать повторно с помощью того же процесса.

Процесс `Reenroll` включает те же процедуры, что и процесс `Enroll`. Исключается только процедура аутентификации. Процедуры `Reenroll` в основном повторяют процедуры `Enroll`. Отличия описываются ниже.

Генерация ключей выполняется также, как в сценарии Enroll1. Если субъект переносит в новый сертификат открытый ключ действующего, то генерация не выполняется.

Запрос на выпуск сертификата готовится также, как в Enroll1. Субъект указывает в запросе либо только что сгенерированный открытый ключ, либо открытый ключ действующего сертификата. Субъект переносит в запрос идентификационные данные из своего действующего сертификата. При переносе субъект может изменить только дополнительные идентификационные атрибуты.

В атрибуте challengePassword запроса субъект может передать УЦ информационную строку (например, об оплате услуг доверия).

Субъект сам заверяет свой запрос, подписывая его на открытом ключе действующего сертификата. Подписанный запрос оформляется как контейнер SignedData, конвертуется на открытом ключе УЦ и отправляется УЦ.

УЦ обрабатывает запрос по алгоритму процесса Enroll с учетом следующих корректировок.

1 На шаге 3.1 алгоритма не проверять, что в расширении CertificatePolicies сертификата отправителя установлена роль РЦ. Вместо этого проверить, что основные идентификационные атрибуты в сертификате отправителя совпадают с основными идентификационными атрибутами в запросе. Основные идентификационные атрибуты определены в 7.3.1.

2 Если в запросе указаны новые дополнительные идентификационные атрибуты, то на шаге 3.1 следует дополнительно проверить, что субъект владеет соответствующими цифровыми ресурсами.

3 На шаге 3.1 дополнительно проверить, что атрибут serialNumber не начинается с префикса IDCBY- и, таким образом, речь не идет о сертификате, выданном с помощью Enroll2.

4 На шаге 3.3 проверить, что в контейнер SignedData вложены данные типа bpki-ct-reenroll-req.

5 Пропустить шаг 4.

6 Если в запросе повторяется открытый ключ сертификата отправителя, то:

1) на шаге 5.4 повторить начало действия сертификата отправителя в выпускаемом сертификате. Повтор означает накопление срока действия открытого ключа при его переносе из сертификата в сертификат;
2) на шаге 5.6 игнорировать рекомендуемое начало действия сертификата.

7 Выполнить дополнительный шаг: отозвать сертификат отправителя запроса. В расширении reasonCode записи об отозванном сертификате (см. 8.5) указать superseded.

По результатам обработки запроса УЦ формирует ответ: новый сертификат или контейнер BPKIResp. Новый сертификат конвертуется на открытом ключе самого сертификата. Контейнер BPKIResp описывает ошибку во время обработки запроса. Контейнер интерпретируется также, как в процессе Enroll. Как и в Enroll контейнер подписывается на ключе агента УЦ.

Ответ УЦ получает субъект. Получатель обрабатывает ответ также, как в процессе Enroll. Если получено сообщение об ошибке со статусом waiting, то субъект должен сохранить идентификатор запроса и уточнить статус позже, выполнив процесс Retrieve.

УЦ должен гарантировать действительность старого сертификата в период сохранения статуса waiting.

### 9.4 Процесс Spawn

Процесс Spawn выполняют субъект сертификата и УЦ. С помощью Spawn субъект получает новый сертификат, подтверждая владение действующим. В новый сертификат переносятся идентификационные данные из действующего. Разрешается изменять только дополнительные идентификационные атрибуты. Действующий сертификат не отзывается.

УЦ должен разработать и реализовать политику продления идентификационных данных в процессе Spawn, аналогичную такой же политике процесса Reenroll.

УЦ, который выдает новый сертификат, может отличаться от УЦ, выдавшего действующий. Например, действующий сертификат может быть временным, выданным корпоративным ПУЦ, а новый сертификат — долгосрочным, выдаваемым РУЦ.

Процесс Spawn включает те же процедуры, что и процесс Reenroll. Процедуры процессов отличаются в следующем.

1 В контейнере SignedData с запросом на получение сертификата тип содержимого меняется на bpki-ct-spawn-req.

2 Генерируется новая пара ключей. Открытый ключ действующего сертификата не переносится в новый.

3 При обработке запроса УЦ пропускает шаг отзыва действующего сертификата.

### 9.5 Процесс Retrieve

Процесс Retrieve выполняют отправитель не до конца обработанного запроса на выпуск сертификата и УЦ. Отправитель — это сторона, которая выполнила один из процессов Enroll, Reenroll или Spawn и по его окончании получила статус waiting. Сохранив идентификатор requestId своего запроса, отправитель с помощью Retrieve может завершить его обработку.

Процесс Retrieve состоит из следующих процедур:

- отправка запроса УЦ;
- обработка запроса;
- возврат ответа;
- обработка ответа.

Запрос УЦ представляет собой контейнер BPKIRetrieveReq, который определен в 8.12. Отправитель записывает в компонент requestId контейнера идентификатор первоначального запроса, а в компонент nonce — случайную строку октетов. Контейнер отправляется УЦ.

УЦ проверяет наличие присланного в контейнере requestId в списке сохраненных пар «идентификатор запроса — сертификат отправителя». Этот список формируется при обработке запросов в процессах Enroll, Reenroll и Spawn.

Если идентификатор не найден, то УЦ возвращает контейнер BPKIResp со статусом rejection. Если идентификатор найден, но выпуск сертификата по соответствующему запросу все еще не завершен, то в контейнере возвращается статус waiting. Если идентификатор найден, но выпуск сертификата завершен с ошибкой, то в контейнере снова возвращается статус rejection. Во всех случаях requestId и nonce из запроса переносятся в соответствующие компоненты BPKIResp.

Ответ BPKIResp подписывается агентом УЦ. Подписанный ответ оформляется как контейнер SignedData.

Если сертификат все-таки выпущен, то УЦ конвертует его на открытом ключе из сертификата отправителя, соответствующего `requestId`. Ответ отправляется в виде контейнера `EnvelopedData`.

Отправитель обрабатывает ответ УЦ так, как если бы он был получен в том процессе, в котором был отправлен первоначальный запрос. Дополнительно отправитель проверяет компонент `nonce` в ответах типа `BPKIResp` — его значение должно совпадать с первоначальной синхропосылкой, выбранной отправителем.

### 9.6 Процесс `Setpwd`

Процесс `Setpwd` выполняют субъект сертификата и УЦ. С помощью `Setpwd` субъект меняет пароль отзыва своего сертификата. Пароль позволяет выполнить отзыв даже при потере соответствующего личного ключа.

При выпуске сертификатов следует предусмотреть информирование субъекта о возможности смены пароля в будущем с помощью процесса `Setpwd`.

Процесс `Setpwd` состоит из следующих процедур:

- подготовка запроса;
- отправка запроса УЦ;
- обработка запроса;
- возврат ответа;
- обработка ответа.

Субъект выбирает пароль, представляет его значением типа `UTF8String` и подписывает это значение на своем личном ключе. Подписанный запрос оформляется как контейнер `SignedData`. Субъекту следует выбирать высокоэнтропийные пароли достаточно большой длины. Паролю должен предшествовать префикс `'/RPWD:'`.

Субъект конвертует подписанный пароль на открытом ключе УЦ и отправляет его УЦ в виде контейнера `EnvelopedData`. Перед отправкой субъект вычисляет идентификатор конвертованного запроса, хэшируя его с помощью `belt-hash`.

УЦ снимает защиту с контейнера `EnvelopedData` и определяет вложенный контейнер `SignedData`. УЦ находит в контейнере сертификат отправителя и проверяет, что сертификат действительно выдан самим УЦ. После этого УЦ проверяет тип содержимого и подпись контейнера `SignedData`. Тип содержимого должен равняться `bpki-ct-setpwd-req`. Если проверки прошли успешно, то УЦ связывает присланный пароль с сертификатом отправителя. УЦ может отказать в связывании, если пароль не удовлетворяет определенным метрикам качества.

Примечание — УЦ может связывать пароль со всеми сертификатами определенного субъекта или даже с самим субъектом как потребителем услуг доверия УЦ.

УЦ возвращает статус обработки запроса в контейнере `BPKIResp`. Разрешены статусы `granted` и `rejection`. В компоненте `requestId` контейнера указывается идентификатор запроса, вычисленный УЦ с помощью `belt-hash`.

Ответ `BPKIResp` подписывается агентом УЦ. Подписанный ответ оформляется как контейнер `SignedData`. Контейнер отправляется субъекту.

Субъект проверяет тип содержимого и подпись ответа. Тип содержимого должен равняться `bpki-ct-resp`. Если подпись действительна, то субъект проверяет совпадение указанного в ответе идентификатора `requestId` с сохраненным идентификатором запроса. При успешной проверке субъект разбирает статус ответа и либо переходит на новый пароль при статусе `granted`, либо оставляет действующий при статусе `rejection`.

При ошибках проверки субъект не может быть уверен, что УЦ обработал его запрос и перешел на новый пароль. Поэтому субъект должен хранить некоторое время два пароля. Окончательное решение о смене пароля субъект должен принять, повторно выполнив `Setpwd`.

### 9.7 Процесс `Revoke`

Процесс `Revoke` выполняют субъект сертификата и УЦ — его эмитент. С помощью `Revoke` субъект отзывает свой сертификат. Отзыв возможен, если субъект сохранил свой личный ключ и может подписать на нем запрос на отзыв, или если субъект предварительно связал с сертификатом пароль отзыва. Пароль отзыва регистрируется и обновляется в процессе `Setpwd`.

Процесс `Revoke` состоит из следующих процедур:

- подготовка запроса на отзыв;
- отправка запроса УЦ;
- обработка запроса;
- возврат ответа;
- обработка ответа.

Запрос на отзыв представляет собой контейнер `BPKIRevokeReq`, который определен в 8.4. Субъект переносит в `BPKIRevokeReq` идентификационные данные эмитента (целевого УЦ) и серийный номер отзываемого сертификата, дает рекомендации по заполнению записи об отзыве сертификата, комментирует причину отзыва. В компоненте `revokePwd` контейнера субъект указывает пароль отзыва. Пароль можно не задавать, если личный ключ не потерян и запрос будет подписываться.

Запрос подписывается обязательно, если пароль отзыва не был зарегистрирован, и по желанию субъекта в противном случае. Рекомендуется подписывать запрос всегда, когда это возможно. Подписанный запрос оформляется как контейнер `SignedData`.

Субъект конвертует запрос или подписанный запрос на открытом ключе УЦ и отправляет его УЦ в виде контейнера `EnvelopedData`. Перед отправкой субъект вычисляет идентификатор конвертованного запроса, хэшируя его с помощью `belt-hash`.

УЦ снимает защиту с контейнера `EnvelopedData` и определяет либо вложенный контейнер `SignedData`, либо непосредственно запрос `BPKIRevokeReq`.

В первом случае УЦ находит в контейнере `SignedData` сертификат отправителя и проверяет, что сертификат действительно был выдан УЦ и не был отозван к текущему моменту времени. После этого УЦ проверяет тип содержимого и подпись контейнера `SignedData`. Тип содержимого должен равняться `bpki-ct-revoke-req`. Наконец УЦ проверяет, что подписан запрос `BPKIRevokeReq` и что компоненты `issuer` и `serialNumber` этого запроса совпадают с одноименными компонентами сертификата отправителя.

Во втором случае УЦ проверяет, что сертификат с реквизитами `issuer` и `serialNumber` из запроса действительно был выдан УЦ, не был отозван и что с этим сертификатом действительно связан указанный в запросе пароль `revokePwd`. УЦ должен организовать защиту от перебора `revokePwd` злоумышленником. Например, УЦ может приостановить прием запросов на отзыв определенного сертификата после нескольких неверных вариантов пароля. Или УЦ может потребовать выполнения вычислительной работы перед отправкой запросов (см. 10.5).

В каждом из случаев при положительном результате всех проверок УЦ отзывает сертификат. При формировании записи об отзыве УЦ может учесть рекомендации отправителя запроса, заданные в компонентах `reasonCode` и `invalidityDate`.

УЦ возвращает статус обработки запроса в контейнере `BPKIResp`. Разрешены статусы `granted` и `rejection`. В компоненте `requestId` контейнера указывается идентификатор запроса, вычисленный УЦ с помощью `belt-hash`.

Ответ `BPKIResp` подписывает агент УЦ. Подписанный ответ оформляется как контейнер `SignedData`. Контейнер отправляется субъекту.

Субъект проверяет тип содержимого и подпись ответа. Тип содержимого должен равняться `bpki-ct-resp`. Если подпись действительна, то субъект проверяет совпадение указанного в ответе идентификатора `requestId` с сохраненным идентификатором запроса. При успешной проверке субъект разбирает статус ответа, т. е. статус отзыва сертификата.

При ошибках проверки ответа субъект не может быть уверен, что УЦ обработал его запрос и отозвал сертификат. Субъект может проверить статус отзыва у OCSP-сервера или по актуальному СОС. Кроме этого, субъект может повторно выполнить `Revoke`.

# 10 Транспорт

## 10.1 Основные положения

Для реализации процессов, определенных в 9, требуется организовать сетевое взаимодействие с УЦ. Взаимодействие носит клиент-серверный характер. УЦ выступает в роли сервера — получает и обрабатывает запросы, высылает ответы. Работу УЦ как сервера поддерживают агенты. Нужен, по крайней мере, один агент, который подписывает ответы формата `BPKIResp` (см. 8.11). Клиентом, в зависимости от процесса, может быть оператор РЦ, сам РЦ или субъект сертификата.

Взаимодействие между клиентом и сервером ведется по протоколу HTTP [2]. Использование определенного в СТБ 34.101.65 протокола TLS, который является стандартным инструментом защиты пакетов HTTP, рекомендуется, так как повышает безопасность, но не является обязательным: необходимый уровень защиты сообщений обеспечивается в самих процессах.

Протокол HTTP рекомендуется также использовать для организации взаимодействия с OCSP-сервером, СШВ и СЗД.

## 10.2 Сетевые узлы

Каждый из процессов, который реализует УЦ, обслуживается определенным сетевым узлом. Имя узла должно представлять собой URI, компонент `path` которого повторяет имя процесса с заменой первой прописной буквы на строчную и добавлением в начало наклонной черты: '`/enroll`', '`/reenroll`', '`/spawn`' и т. д.

Для обслуживания сценариев процесса `Enroll` могут вводиться отдельные сетевые узлы. Имена этим узлам назначаются по аналогичным правилам: компонент `path` имеют вид '`/enroll1`', '`/enroll2`' или '`/enroll3`'.

При назначении имен узлам OCSP-серверов, СШВ и СЗД в компонентах `path` рекомендуется соответственно указывать '`/ocsp`', '`/tsa`' и '`/dvcs`'.

Имена узлов, поддерживаемых УЦ, серверами и службами, могут указываться в расширении `SubjectAltName` их собственных сертификатов или сертификатов их агентов.

Примеры имен узлов: '`http://bpki.by/setpwd`', '`http://bpki.by/enroll3`', '`http://bpki.by/tsa`'.

### 10.3 Пакеты

Запрос HTTP-серверу и соответствующий ответ представляют собой пакеты, формат которых определен в [2]. Запрос должен отправляться с помощью метода `POST`.

Заголовки пакетов СШВ и СЗД настраиваются по правилам, определенным в СТБ 34.101.81 и СТБ 34.101.82.

Дополнительные правила:

1 В запросах и ответах УЦ, при обслуживании им процессов раздела 9, заголовок `Content-Type` должен принимать значение '`application/cms`'.

2 В запросе OCSP-серверу заголовок `Content-Type` должен принимать значение '`application/ocsp-request`'. В соответствующем ответе этот же заголовок должен принимать значение '`application/ocsp-response`'.

В теле пакета передается контейнер ACH.1. Контейнер должен кодироваться по правилам DER, DER-код должен представляться в двоичном виде.

### 10.4 Коды ответов

HTTP-код ответа должен выбираться в соответствии с [2]: 200 — в случае успешной обработки, 202 — в случае отложенной обработки, коды 400-499 — в случае отказа в проведении операции, коды 500-599 — в случае непредвиденных ошибок HTTP-сервера.

Код ответа должен соответствовать содержимому ответа, т. е. содержимому вложенного в ответ контейнера ACH.1.

### 10.5 Заголовок `Nonce`

При обработке входящих запросов HTTP-серверу приходится выполнять достаточно трудоемкие вычисления, связанные с проверкой ЭЦП и снятием защиты с конвертованных данных. Злоумышленник может навязать серверу обработку большого числа пакетов, заблокировав тем самым прием данных от других сторон. Кроме этого, злоумышленник может отправлять УЦ на узел '`/revoke`' большое число пакетов со ссылкой на сертификат другой стороны и вариантами пароля его отзыва, ожидая, что один из вариантов подойдет и сертификат будет несанкционированно отозван.

Для защиты от перечисленных угроз можно затруднить отправку пакетов серверу, навязывая клиенту вычислительную работу в момент отправки. Для этого рекомендуется использовать HTTP-заголовок `Nonce`, определяемый ниже.

В запросе HTTP-серверу заголовок `Nonce` содержит синхропосылку $S \in \{0,1\}^{64}$ и натуральное число $d$, называемое уровнем (вычислительной работы). Синхропосылка представляется строкой $\mathtt{hex}(S)$, уровень $d$ — строкой его десятичных цифр. Строки разделяются двоеточием.

Пример заголовка в запросе серверу: '`Nonce: 0123456789ABCDEF:16`'.

При использовании заголовка `Nonce` пакет должен дополнительно содержать заголовок `Date`, в котором фиксируется время отправки пакета. Время задается текстовой строкой $t$, формат которой определен в [2]. Эта строка представляется словом $T \in \{0,1\}^{8*}$ таким, что $\mathtt{hex}(T) = t$.

Уровень $d$ характеризует объем вычислений, которые были проведены с телом пакета $B \in \{0,1\}^{8*}$ относительно момента времени $T$. В вычислениях участвует функция хэширования `belt-hash` (см. 6.2). Отправитель пакета:

1) фиксирует текущий момент времени $T$;
2) выбирает случайную синхропосылку $S$;

3) вычисляет хэш-значение $H = \texttt{belt-hash}(S \parallel T \parallel B)$;

4) если слово $H$ не начинается с $d$ нулей, то возвращается к шагу 1.

До обнаружения подходящего $H$ отправителю в среднем требуется выполнить $2^d$ хэширований, т. е. провести вычислительную работу объема порядка $2^d$. Серверу для проверки работы нужно выполнить всего одно хэширование.

Сервер отказывается от дальнейшей обработки пакета, если:

1) время в заголовке `Date` задано некорректно или значительно отличается от текущего;

2) синхропосылка $S$ в заголовке `Nonce` задана некорректно или уровень $d$ недостаточно большой;

3) хэш-значение $\texttt{belt-hash}(S \parallel T \parallel B)$ не начинается с $d$ нулей.

Сервер заранее информирует клиентов о минимально допустимом уровне $d$. Или сервер сообщает об этом уровне в своем ответе, информирующем об отказе в обработке. Код ответа должен быть 408 в случае нарушения первого условия, или 428 в случае нарушения двух последних.

При возврате кода 428 сервер должен включить в ответ заголовок `Nonce` и в нем указать минимально допустимый уровень $d$.

Пример заголовка в ответе сервера: '`Nonce: 20`'.

# 11 Формат ключевого контейнера

## 11.1 Правила защиты

Контейнер программного КТ содержит личный ключ СТБ 34.101.45. Контейнер может защищаться на пароле — строке в естественном алфавите. Строку должен суметь запомнить владелец личного ключа, и поэтому пароль обычно имеет сравнительно небольшую длину, низкую энтропию, его определение путем перебора намного проще, чем определение личного ключа с помощью известных криптоаналитических атак.

Чтобы сохранить стойкость личного ключа при хранении, контейнер должен дополнительно защищаться аппаратно. Например, контейнер может храниться внутри устройства, доступ к объектам которого осуществляется только после предъявления пароля, причем доступ блокируется после нескольких неверных попыток ввода пароля. Для сравнения, блокировать ввод пароля при прямом доступе к контейнеру нельзя.

Если дополнительная защита контейнера затруднена, то вместо низкоэнтропийного пароля следует использовать высокоэнтропийный ключ — случайное двоичное слово $P$ длины $l \in \{128, 192, 256\}$, где $l$ — уровень стойкости личного ключа. Ключ защиты следует разделить на несколько частичных секретов с помощью алгоритма, определенного в СТБ 34.101.60 (п. 7.3).

Частичные секреты также являются двоичными словами длины $l$. Их количество не должно превышать 16. Пороговое число (минимальное число частичных секретов, необходимых для восстановления ключа) должно быть не меньше 2. При разделении должны использоваться стандартные открытые ключи, заданные в СТБ 34.101.60 (приложение А). При этом с каждым частичным секретом связывается определенный открытый ключ одной из таблиц А.2, А.3, А.4 приложения. Номер открытого ключа считается номером частичного секрета. Номер кодирует октетом: октет $\texttt{01}_{16}$ представляет номер 1, октет $\texttt{02}_{16}$ — номер 2,..., октет $\texttt{10}_{16}$ — номер 16. Октет номера записывается в начало частичного секрета и хранится вместе с ним.

Частичные секреты размещаются на различных носителях информации владельца, в нескольких его сетевых репозиториях. Размещение должно быть организовано так, чтобы несанкционированный доступ к пороговому числу частичных секретов был максимально затруднен.

Частичные секреты могут сохраняться во вспомогательных контейнерах, защищенных на паролях. Пароли защиты различных контейнеров могут совпадать. В любой комбинации порогового числа частичных секретов хотя бы один из них должен быть защищен на пароле.

Ключ $P$ защиты контейнера с личным ключом кодируется текстовой строкой $\texttt{hex}(P)$ длины $l/4$. Таким образом, для защиты контейнера любого типа всегда используется секретная строка: либо пароль, либо закодированный ключ.

Частичный секрет, который хранится в открытом виде, также может кодироваться текстовой строкой. Эта строка будет иметь длину $l/4 + 2$. Первые два символа строки представляют номер секрета.

Формат ключевого контейнера описывается типом `EncryptedPrivateKeyInfo`, определенным в 11.5. Контейнер `EncryptedPrivateKeyInfo` содержит защищенный вложенный контейнер, в котором непосредственно хранится личный ключ или частичный секрет (см. рисунок 2). Формат вложенного контейнера описывается типом `PrivateKeyInfo`, определенным в 11.4.

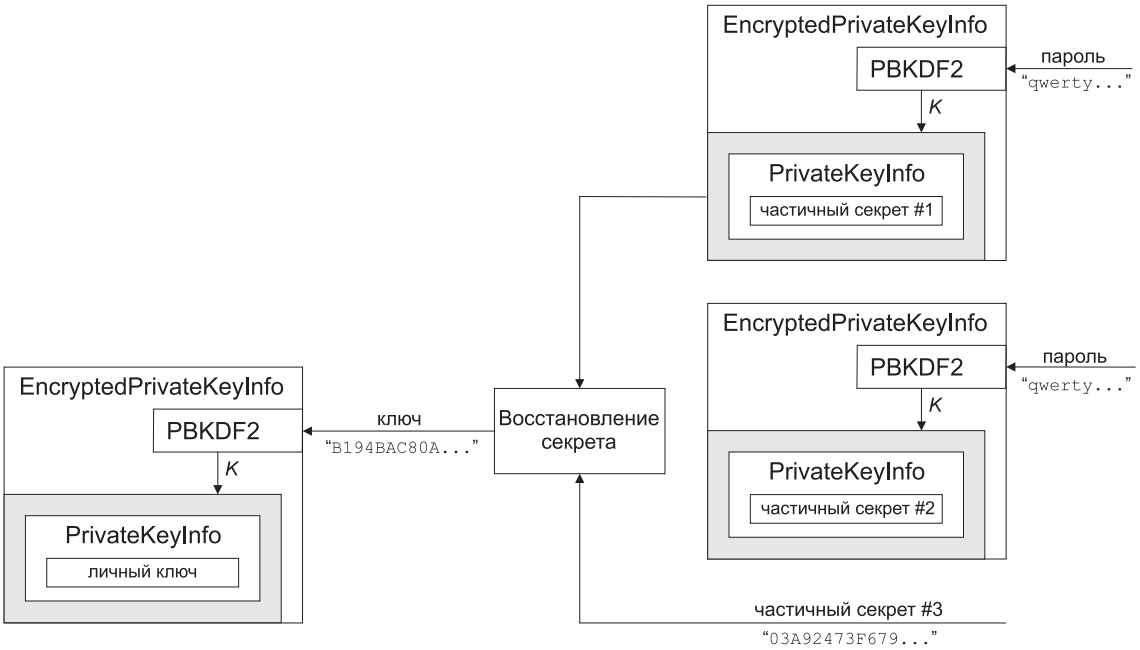


**Рисунок 2 — Защита ключевого контейнера**

### 11.2 Установка защиты

Контейнер `PrivateKeyInfo` защищается на секретной строке и встраивается в контейнер `EncryptedPrivateKeyInfo` следующим образом.

1 Входная секретная строка кодируется строкой октетов по правилам UTF-8 [9].

2 По полученной строке строится ключ $K \in \{0,1\}^{256}$. Для этого используется алгоритм PBKDF2, определенный в [10] и конкретизированный в СТБ 34.101.45 (п. E.2).

3 Контейнер `PrivateKeyInfo` кодируется по базовым правилам (BER). В результате кодирования получается строка октетов $X \in \{0,1\}^{8*}$.

4 Строка $X$ защищается с помощью алгоритма `belt-keywrap`, определенного в СТБ 34.101.31 (шифрование и имитозащита ключа, установка защиты). На вход алгоритма подаются $X$, нулевой заголовок $I \in \{0,1\}^{128}$ и ключ $K$, построенный на шаге 2. Алгоритм возвращает защищенную строку $Y \in \{0,1\}^{|X|+128}$.

5 Строка $Y$ записывается в компонент `encryptedData` контейнера `EncryptedPrivateKeyInfo` (см. 11.5). Остальные компоненты контейнера заполняются служебной информацией и параметрами PBKDF2.

### 11.3 Снятие защиты

Снятие защиты с контейнера `EncryptedPrivateKeyInfo` означает определение вложенного контейнера `PrivateKeyInfo` в открытом виде. Защита снимается на той же секретной строке, которая использовалась при установке защиты. Снятие защиты выполняется следующим образом.

1 Входная секретная строка кодируется строкой октетов по правилам UTF-8 [9].

2 По полученной строке с помощью алгоритма PBKDF2 вычисляется ключ $K \in \{0,1\}^{256}$. Параметры PBKDF2 предварительно считываются из контейнера.

3 Определяется строка октетов $Y \in \{0,1\}^{8*}$, размещенная в компоненте `encryptedData` контейнера.

4 Строка $Y$ обрабатывается алгоритмом $\texttt{belt-keywrap}^{-1}$, определенным в СТБ 34.101.31 (шифрование и имитозащита ключа, снятие защиты). На вход алгоритма подаются $Y$, нулевой заголовок $I \in \{0,1\}^{128}$ и ключ $K$, построенный на шаге 2. Алгоритм возвращает открытую строку $X \in \{0,1\}^{|Y|-128}$ или признак `ОШИБКА`. Возврат признака `ОШИБКА` означает, что либо нарушена целостность контейнера, либо входная секретная строка неверна. При возврате признака снятие защиты преждевременно завершается с ошибкой.

5 Строка $X$ интерпретируется как BER-код контейнера `PrivateKeyInfo`. Строка декодируется, определяется вложенный контейнер.

Снятие защиты завершается с ошибкой не только на шаге 4, но и на шагах 2, 3, 5 при несоблюдении форматов `EncryptedPrivateKeyInfo` и `PrivateKeyInfo`.

### 11.4 Тип `PrivateKeyInfo`

Тип `PrivateKeyInfo` определен в [11]. В настоящем стандарте он конкретизируется следующим образом.

```
PrivateKeyInfo ::= SEQUENCE {
  version                   INTEGER(0),
  keyAlgorithm              CHOICE {
    bignPrivkeyAlgorithm    BignAlgorithmIdentifier,
    belsSharekeyAlgorithm   BelsAlgorithmIdentifier },
  key                       OCTET STRING }

BignAlgorithmIdentifier ::= SEQUENCE {
  algorithm  OBJECT IDENTIFIER(bign-pubkey),
  params     OBJECT IDENTIFIER(bign-curve256v1 | bign-curve384v1 |
                               bign-curve512v1) }

BelsAlgorithmIdentifier ::= SEQUENCE {
```

```
  algorithm  OBJECT IDENTIFIER(bels-share),
  params     OBJECT IDENTIFIER(bels-m0128v1 | bels-m0192v1 | bels-m0256v1) }
```

Компонент `keyAlgorithm` выбирается двумя способами: выбор `bignPrivkeyAlgorithm` означает, что в контейнере хранится личный ключ, выбор `belsSharekeyAlgorithm` — частичный секрет.

Компонент `bignAlgorithm` идентифицирует личный ключ и параметры эллиптической кривой, к которым он привязан. Могут использоваться только стандартные параметры, определенные в СТБ 34.101.45 (приложение Б). Задействованные идентификаторы `bign-pubkey`, `bign-curve256v1`, `bign-curve384v1` и `bign-curve512v1` определены в СТБ 34.101.45 (приложение Д).

Аналогичным образом компонент `belsAlgorithm` идентифицирует частичный секрет и долговременный общий открытый ключ. Могут использоваться только стандартные общие ключи, определенные в СТБ 34.101.60 (приложение А). Идентификаторы `bels-share`, `bels-m0128v1`, `bels-m0192v1`, `bels-m0256v1` определены в СТБ 34.101.60 (приложение Г).

Личный ключ или частичный секрет хранятся в компоненте `key` в виде строки октетов. Длина строки должна соответствовать уровню стойкости $l$, который определяется по стандартным параметрам (эллиптической кривой или общему открытому ключу).

Строка личного ключа строится по правилам СТБ 34.101.45 (п. 5.4). Строка состоит из $l/4$ октетов. Строка частичного секрета состоит из $l/8 + 1$ октетов. Первый октет представляет номер частичного секрета (см. 11.1).

### 11.5 Тип `EncryptedPrivateKeyInfo`

Тип `EncryptedPrivateKeyInfo` определен в [10]. В настоящем стандарте он конкретизируется следующим образом.

```
EncryptedPrivateKeyInfo ::= SEQUENCE {
  encryptionAlgorithm  EncryptionAlgorithmIdentifier,
  encryptedData        OCTET STRING }

EncryptionAlgorithmIdentifier ::= SEQUENCE {
  algorithm   OBJECT IDENTIFIER(id-PBES2),
  params      PBES2-params }

PBES2-params ::= SEQUENCE {
  keyDerivationFunc PBKDF2AlgorithmIdentifier,
  encryptionScheme  BeltKeywrapAlgorithmIdentifier }

PBKDF2AlgorithmIdentifier ::= SEQUENCE {
  algorithm   OBJECT IDENTIFIER(id-PBKDF2),
  params      PBKDF2-params }

BeltKeywrapAlgorithmIdentifier ::= SEQUENCE {
  algorithm   OBJECT IDENTIFIER(belt-keywrap256),
  params      NULL }

PBKDF2-params ::= SEQUENCE {
```

```
  salt            OCTET STRING(SIZE(8)),
  iterationCount  INTEGER (10000..MAX),
  prf             PrfAlgorithmIdentifier }

PrfAlgorithmIdentifier ::= SEQUENCE {
  algorithm OBJECT IDENTIFIER(hmac-hbelt),
  params    NULL }
```

Компонент `encryptedData` типа `EncryptedPrivateKeyInfo` представляет защищенный вложенный контейнер, а компонент `encryptionAlgorithm` описывает алгоритмы защиты и их параметры.

Задействованные в описаниях идентификаторы `id-PBES2` и `id-PBKDF2` определены в [10], а также в СТБ 34.101.45 (приложение Е). Идентификатор `belt-keywrap256` алгоритмов `belt-keywrap`, `belt-keywrap`$^{-1}$ с 256-битовыми ключами определен в СТБ 34.101.31 (приложение Б).

Тип `PBKDF2-params` описывает параметры PBKDF2. Введенные в описание ограничения соответствуют рекомендациям СТБ 34.101.45 (п. Е.4). В частности, для имитозащиты должен использоваться алгоритм HMAC, определенный в СТБ 34.101.47 (п. 6.1), с функцией хэширования, определенной в СТБ 34.101.31 (п. 6.9). Идентификатор `hmac-hbelt` этого алгоритма определен в СТБ 34.101.47 (приложение Б).

# 12 Программный интерфейс

## 12.1 Общие положения

PKCS#11, принятый как СТБ 34.101.21, определяет программный интерфейс взаимодействия с КТ. Интерфейс, называемый Cryptoki, определяет фиксированный набор функций языка Си, позволяющий выполнять широкий набор криптографических операций.

Cryptoki оперирует такими понятиями, как ключевой объект и механизм. В настоящем разделе уточняется использование введенных в СТБ 34.101.21 объектов и вводятся новые механизмы для поддержки алгоритмов СТБ 34.101.45.

Уточняются объект открытого ключа (см. 12.2.2) и объект личного ключа (см. 12.2.3). Объекты генерируются с помощью стандартного механизма СТБ 34.101.21 (см. 12.3.1). Сгенерированный личный ключ сохраняется внутри КТ, открытый ключ экспортируется за пределы КТ с последующим переносом в сертификат. Объекты используются в механизмах ЭЦП (см. 12.3.2 — 12.3.4) и транспорта ключа (см. 12.3.5). В этих механизмах поддержка операций с открытым ключом (проверка ЭЦП и создание токена ключа) не является обязательной, операции можно выполнить за пределами КТ. Одна и та же пара ключей может использоваться в нескольких механизмах.

## 12.2 Объекты

### 12.2.1 Параметры эллиптической кривой

В объектах открытого и личного ключа указывается идентификатор параметров ЭК, с которыми связан ключ объекта. Должен использоваться идентификатор из перечня, заданного в 6.1. Идентификатор должен кодироваться по правилам DER и указываться в атрибуте `CKA_EC_PARAMS`.

Параметры ЭК однозначно определяют длины личного и открытого ключей, подписываемого хэш-значения и подписи: $l/4$, $l/2$, $l/4$ и $3l/8$ октетов соответственно, где $l$ — уровень стойкости (см. 6).

Параметры ЭК используются во всех определяемых далее криптографических механизмах. Этот факт должен быть зафиксирован в структуре `CK_MECHANISM_INFO`, описывающей механизм, через установку в поле `flags` следующих флагов:

– `CKF_EC_F_P` — используется ЭК над простым полем;
– `CKF_EC_NAMEDCURVE` — параметры ЭК задаются идентификатором;
– `CKF_EC_UNCOMPRESS` — точки ЭК задаются в несжатом виде.

### 12.2.2 Объект открытого ключа

Атрибуты объекта открытого ключа СТБ 34.101.45 выбираются и настраиваются по правилам СТБ 34.101.21 со следующими уточнениями.

1 Атрибут `CKA_CLASS` должен принимать значение `CKO_PUBLIC_KEY`.

2 Атрибут `CKA_KEY_TYPE` должен принимать значение `CKK_EC`.

3 Атрибут `CKA_DERIVE` должен принимать значение `CK_FALSE`.

4 Атрибут `CKA_TOKEN` должен принимать значение `CK_TRUE`, если ключ является объектом КТ и доступен всем приложениям, подключенным к КТ. Иначе ключ считается сеансовым объектом, который доступен только приложению, создавшему его, и который удаляется автоматически при закрытии сеанса.

5 В атрибуте `CKA_EC_PARAMS` должен быть задан идентификатор связанных параметров ЭК (см. 12.2.1).

6 В атрибуте `CKA_EC_POINT` задается значение открытого ключа. Атрибут должен использоваться для создания объекта открытого ключа по значению открытого ключа либо для извлечения значения из объекта.

7 В атрибуте `CKA_ID` задается идентификатор открытого ключа. Атрибут должен использоваться в тех случаях, когда на КТ хранятся несколько открытых ключей, и требуется выбирать один из них. Атрибут `CKA_ID` должен быть согласован с одноименным атрибутом объекта личного ключа.

8 Атрибут `CKA_VERIFY` должен принимать значение `CK_TRUE` только если открытый ключ планируется использовать для проверки ЭЦП.

9 Атрибут `CKA_WRAP` должен принимать значение `CK_TRUE` только если открытый ключ планируется использовать для создания токена ключа.

10 В атрибуте `CKA_ALLOWED_MECHANISMS` должен быть указан список идентификаторов механизмов, с которыми разрешается использовать открытый ключ. Список должен быть согласован с уровнем стойкости ключа и со значениями атрибутов `CKA_VERIFY`, `CKA_WRAP`. Атрибут может указываться при создании объекта открытого ключа или генерации пары ключей для ограничения использования открытого ключа.

Ниже приведен пример шаблона для генерации объекта открытого ключа СТБ 34.101.45.

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_UTF8CHAR label[] = "bign128-pubkey";
CK_BYTE bignParams[] = {
  0x06,0x0A,0x2A,0x70,0x00,0x02,
  0x00,0x22,0x65,0x2D,0x03,0x01};
```

```
CK_BYTE id[] = {1};
CK_MECHANISM_TYPE mechanisms[] = {
  CKM_BIGN, CKM_BIGN_TSP };
CK_BBOOL true = CK_TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label) - 1},
  {CKA_EC_PARAMS, bignParams, sizeof(bignParams)},
  {CKA_ID, id, sizeof(id)},
  {CKA_VERIFY, &true, sizeof(true)},
  {CKA_WRAP, &true, sizeof(true)},
  {CKA_ALLOWED_MECHANISMS, mechanisms, sizeof(mechanisms)},
};
```

### 12.2.3 Объект личного ключа

Атрибуты объекта личного ключа СТБ 34.101.45 выбираются и настраиваются по правилам СТБ 34.101.21 со следующими уточнениями.

1 Атрибут `CKA_CLASS` должен принимать значение `CKO_PRIVATE_KEY`.

2 Атрибут `CKA_KEY_TYPE` должен принимать значение `CKK_EC`.

3 Атрибут `CKA_DERIVE` должен принимать значение `CK_FALSE`.

4 Атрибут `CKA_TOKEN` должен принимать значение `CK_TRUE`, если ключ является объектом КТ и доступен всем приложениям, подключенным к КТ. Иначе ключ считается сеансовым объектом, который доступен только приложению, создавшему его, и который удаляется автоматически при закрытии сеанса.

5 В атрибуте `CKA_EC_PARAMS` должен быть задан идентификатор связанных параметров ЭК (см. 12.2.1).

6 В атрибуте `CKA_VALUE` задается значение личного ключа. Атрибут может использоваться для ввода / вывода сеансового личного ключа.

7 В атрибуте `CKA_ID` задается идентификатор личного ключа. Атрибут должен использовать в тех случаях, когда в КТ хранятся несколько личных ключей, и требуется выбирать один из них. Значение атрибута `CKA_ID` запрещается изменять после его назначения объекту личного ключа.

8 Атрибут `CKA_SIGN` должен принимать значение `CK_TRUE` только если личный ключ планируется использовать для выработки ЭЦП.

9 Атрибут `CKA_UNWRAP` должен принимать значение `CK_TRUE` только если личный ключ планируется использовать для разбора токена ключа.

10 В атрибуте `CKA_ALLOWED_MECHANISMS` должен быть указан список идентификаторов механизмов, с которыми разрешается использовать личный ключ. Список должен быть согласован с уровнем стойкости ключа и со значениями атрибутов `CKA_SIGN`, `CKA_UNWRAP`. Атрибут может указываться при создании объекта личного ключа или генерации пары ключей для ограничения использования личного ключа.

11 Атрибут `CKA_SENSITIVE` должен принимать значение `CK_TRUE`, если ключ не может быть выведен за пределы КТ ни в каком виде. Именно такое значение должен принимать атрибут, если речь идет о личном ключе токена (см. описание `CKA_TOKEN`).

12 Атрибут CKA_EXTRACTABLE должен принимать значение CK_TRUE, если личный ключ может быть выведен за пределы КТ с помощью транспорта ключа. Атрибут должен принимать значение CK_FALSE или отсутствовать, если речь идет о личном ключе токена.

Ниже приведен пример шаблона для генерации объекта личного ключа СТБ 34.101.45. Этот же шаблон может быть использован для поиска личного ключа.

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_BYTE id[] = {1};
CK_UTF8CHAR label[] = "bign128-privkey";
CK_MECHANISM_TYPE mechanisms[] = {
  CKM_BIGN, CKM_BIGN_TSP };
CK_BBOOL true = CK_TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_ID, id, sizeof(id)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_SENSITIVE, &true, sizeof(true)},
  {CKA_EXTRACTABLE, &false, sizeof(false)},
  {CKA_LABEL, label, sizeof(label) - 1},
  {CKA_SIGN, &true, sizeof(true)},
  {CKA_UNWRAP, &true, sizeof(true)},
  {CKA_ALLOWED_MECHANISMS, mechanisms, sizeof(mechanisms)},
};
```

### 12.3 Механизмы

#### 12.3.1 Механизм CKM_EC_KEY_PAIR_GEN

Для генерации ключей СТБ 34.101.45 должен использоваться стандартный механизм CKM_EC_KEY_PAIR_GEN. Механизм не имеет параметров.

Механизм поддерживается функцией C_GenerateKeyPair. Описатель механизма и шаблоны объектов открытого и личного ключей подаются на вход C_GenerateKeyPair. Параметры ЭК задаются в атрибуте CKA_EC_PARAMS шаблона открытого ключа.

Механизм однозначно определяет атрибуты CKA_CLASS, CKA_KEY_TYPE и CKA_EC_POINT объекта открытого ключа и атрибуты CKA_CLASS, CKA_KEY_TYPE, CKA_EC_PARAMS и CKA_VALUE объекта личного ключа, поэтому указанные атрибуты могут не указываться в соответствующих шаблонах.

#### 12.3.2 Механизм CKM_BIGN

Для выработки и проверки ЭЦП в соответствии с СТБ 34.101.45 по заданному хэш-значению сообщения должен использоваться механизм CKM_BIGN. Параметром механизма (указывается в описателе механизма — структуре CK_MECHANISM) является DER-код идентификатора используемого алгоритма хэширования.

Идентификатор механизма: 0x00008001 (синтаксис языка Си, принятый в СТБ 34.101.21).

Механизм поддерживается функциями выработки и проверки ЭЦП: C_SignInit, C_Sign, C_VerifyInit, C_Verify.

При выработке подписи описатели сеанса, механизма и объекта личного ключа подаются на вход функции C_SignInit. Описатель сеанса, указатели на подписываемое хэш-значение и выходное значение подписи вместе с их размерами подаются на вход функции C_Sign.

При проверке подписи описатели сеанса, механизма и объекта открытого ключа подаются на вход функции C_VerifyInit. Описатель сеанса, указатели на подписанное хэш-значение и значение подписи вместе с их размерами подаются на вход функции C_Verify.

Длины входных и выходных данных (хэш-значение, подпись) функций C_Sign и C_Verify должны соответствовать уровню стойкости $l$ соответствующего ключевого объекта.

Могут поддерживаться только определенные алгоритмы хэширования. При отсутствии поддержки функции C_SignInit, C_VerifyInit должны возвращать код CKR_MECHANISM_PARAM_INVALID.

Могут поддерживаться только алгоритмы выработки ЭЦП. При отсутствии поддержки алгоритмов проверки подписи функция C_VerifyInit должна возвращать код CKR_FUNCTION_NOT_SUPPORTED.

### 12.3.3 Механизм CKM_BIGN_HBELT

Для выработки и проверки ЭЦП в соответствии с алгоритмами bign-with-hbelt (см. 6.4) должен использоваться механизм CKM_BIGN_HBELT. Механизм не имеет параметров.

Идентификатор механизма: 0x00008002.

Механизм поддерживается функциями выработки и проверки ЭЦП: C_SignInit, C_SignUpdate, C_SignFinal, C_VerifyInit, C_VerifyUpdate, C_VerifyFinal.

При выработке подписи описатели сеанса, механизма и объекта личного ключа подаются на вход функции C_SignInit. Описатель сеанса, подписываемые данные целиком или по частям подаются на вход функции C_SignUpdate. Описатель сеанса, указатель на выходное значение подписи вместе с размером подаются на вход функции C_SignFinal.

При проверке подписи описатели сеанса, механизма и объекта открытого ключа подаются на вход функции C_VerifyInit. Описатель сеанса, подписанные данные целиком или по-частям подаются на вход функции C_VerifyUpdate. Описатель сеанса, указатель на значение подписи вместе с размером подаются на вход функции C_VerifyFinal.

В механизме должны использоваться ключи уровня стойкости $l = 128$ и стандартные параметры ЭК этого уровня (см. 6.1). Буфер подписи, подаваемый на вход функций C_SignFinal и C_VerifyFinal, должен состоять из 48 октетов.

Может поддерживаться только алгоритм выработки ЭЦП. При отсутствии поддержки алгоритма проверки подписи функция C_VerifyInit должна возвращать код CKR_FUNCTION_NOT_SUPPORTED.

### 12.3.4 Механизм CKM_BIGN_BASH

Для выработки и проверки ЭЦП в соответствии с алгоритмами bign-with-bash384, bign-with-bash512 (см. 6.4) должен использоваться механизм CKM_BIGN_BASH. Механизм не имеет параметров.

Идентификатор механизма: 0x00008003.

Механизм CKM_BIGN_BASH поддерживается теми же функциями и по тем же правилам, что и механизм CKM_BIGN_HBELT.

В механизме должны использоваться ключи уровней стойкости $l = 192$ и $l = 256$ и стандартные параметры ЭК этих уровней (см. 6.1). Буфер подписи, подаваемый на вход функций `C_SignFinal` и `C_VerifyFinal`, должен состоять из 72 ($l = 192$) или 96 ($l = 256$) октетов.

Механизм может дополнительно реализовывать алгоритмы `bign-with-bash256`: связку алгоритмов ЭЦП СТБ 34.101.45 с алгоритмом хэширования `bash256`, определенным в СТБ 34.101.77. При этом должны использоваться ключи уровня стойкости $l = 256$ и стандартные параметры ЭК этого уровня. Буфер подписи должен состоять из 48 октетов.

### 12.3.5 Механизм CKM_BIGN_TSP

Для разбора и создания токена ключа в соответствии с алгоритмами `bign-keytransport` (см. 6.5) должен использоваться механизм `CKM_BIGN_TSP`. Параметр механизма – заголовок транспортируемого ключа (16 октетов). Параметр может опускаться, и тогда должен использоваться заголовок из 16 нулевых октетов.

Идентификатор механизма: `0x00008004`.

Механизм подерживается функциями создания и разбора токена ключа: `C_WrapKey` и `C_UnwrapKey`.

При создании токена на вход функции `C_WrapKey` подаются описатель сеанса, описатель механизма, описатели объектов открытого ключа получателя и транспортируемого ключа, указатель на выходное значение токена ключа вместе с размером. Размер транспортируемого ключа должен быть не менее 16 октетов. Объект транспортируемого ключа может иметь произвольный класс и тип.

При разборе токена на вход функции `C_UnwrapKey` подаются описатель сеанса, описатель механизма, описатель объекта личного ключа получателя, указатель на значение токена ключа вместе с размером, набор атрибутов для создания нового ключа и указатель, который на выходе получает описатель на созданный ключ. Размер токена ключа должен быть не менее 32 октетов. Объект нового ключа может иметь произвольный класс и тип.

Может поддерживаться только алгоритм разбора токена. При отсутствии поддержки алгоритма создания токена функция `C_WrapKey` должна возвращать код `CKR_FUNCTION_NOT_SUPPORTED`.

## 12.4 Программная библиотека

Программная библиотека, реализующая интерфейс Cryptoki, дополнительно к описанным выше функциям криптографических механизмов должна включать следующие:

- `C_Initialize` — инициализация библиотеки;
- `C_GetInfo` — получение информации о библиотеке;
- `C_GetFunctionList` — получение указателей на реализации функций;
- `C_GetSlotList` — перечисление слотов в системе;
- `C_GetSlotInfo` — получение информации о слоте;
- `C_GetTokenInfo` — получение информации о КТ, находящемся в выбранном слоте;
- `C_GetMechanismList` — получение списка механизмов, поддерживаемых КТ;
- `C_OpenSession` — открытие сеанса работы с КТ;
- `C_Login` — парольная аутентификация пользователя перед КТ;
- `C_FindObjectsInit`, `C_FindObjects`, `C_FindObjectsFinal` — поиск объекта на основании шаблона;
- `C_GetAttributeValue` — получение атрибутов объекта;

– `C_Logout` — завершение работы с критическими объектами КТ;
– `C_CloseSession` — завершение сеанса работы с КТ;
– `C_Finalize` — завершение работы с библиотекой.

Функции должны быть реализованы в соответствии с СТБ 34.101.21. Схема работы с функциями также определена в СТБ 34.101.21.

Параметром функции `C_Login` является пароль владельца КТ. Пароль может содержать любые символы UTF-8, кроме ':' (графический код байта $58 = \mathtt{3A}_{16}$). Пароль может сопровождаться служебной информацией: тип пароля, настройки состояния владельца, разрешения владельца на доступ к объектам. Пример пароля вместе со служебной информацией: '`1123581321:PUK`'.

Экспорт открытого ключа может быть выполнен с помощью функции `C_GetAttributeValue`. В шаблоне запрашиваемых атрибутов, передаваемом на вход функции, следует указать `CKA_EC_POINT`.

# Приложение А
(рекомендуемое)
# Модуль АСН.1

```
Bpki-module-v1 {iso(1) member-body(2) by(112) 0 2 0 34 101 78 module(1) ver1(1)}
DEFINITIONS ::=
BEGIN
  IMPORTS
    CRLReason
      FROM PKIX1Explicit88 {iso(1) identified-organization(3)
        dod(6) internet(1) security(5) mechanisms(5) pkix(7)
        id-mod(0) id-pkix1-explicit-88(1)}
    PKIStatusInfo
      FROM PKIXTSP {iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}
    belt-keywrap256
      FROM Belt-module-v1 {iso(1) member-body(2) by(112) 0 2 0 34 101 31 1 1}
    bign-pubkey, bign-curve256v1, bign-curve384v1, bign-curve512v1
      FROM Bign-module-v2 {iso(1) member-body(2) by(112) 0 2 0 34 101 45 1 2}
    hmac-hbelt
      FROM Brng-module-v2 {iso(1) member-body(2) by(112) 0 2 0 34 101 47 1 2}
    bels-share, bels-m0128v1, bels-m0192v1, bels-m0256v1
      FROM Bels-module-v2 {iso(1) member-body(2) by(112) 0 2 0 34 101 60 1 2}
    id-PBKDF2, id-PBES2
      FROM PKCS5v2-1 {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
        pkcs-5(5) modules(16) pkcs5v2-1(2)}

  bpki OBJECT IDENTIFIER ::= {iso(1) member-body(2) by(112) 0 2 0 34 101 78}

  bpki-role OBJECT IDENTIFIER ::= {bpki 2}

  bpki-role-ca0 OBJECT IDENTIFIER ::= {bpki-role 0}
  bpki-role-ca1 OBJECT IDENTIFIER ::= {bpki-role 1}
  bpki-role-ca2 OBJECT IDENTIFIER ::= {bpki-role 2}
  bpki-role-aa OBJECT IDENTIFIER ::= {bpki-role 10}
  bpki-role-ra OBJECT IDENTIFIER ::= {bpki-role 20}
  bpki-role-ocsp OBJECT IDENTIFIER ::= {bpki-role 30}
  bpki-role-tsa OBJECT IDENTIFIER ::= {bpki-role 31}
  bpki-role-dvcs OBJECT IDENTIFIER ::= {bpki-role 32}
  -- identification servers
  bpki-role-ids OBJECT IDENTIFIER ::= {bpki-role 33}
  bpki-role-tls OBJECT IDENTIFIER ::= {bpki-role 50}
  -- natural persons
  bpki-role-np OBJECT IDENTIFIER ::= {bpki-role 60}
  -- foreign natural persons
```

```
bpki-role-fnp OBJECT IDENTIFIER ::= {bpki-role 61}
-- legal representatives
bpki-role-lr OBJECT IDENTIFIER ::= {bpki-role 62}
-- autonomous cryptographic devices
bpki-role-acd OBJECT IDENTIFIER ::= {bpki-role 70}

-- extended key usage
bpki-eku OBJECT IDENTIFIER ::= {bpki 3}

--  server of Terminal Mode
bpki-eku-serverTM OBJECT IDENTIFIER ::= {bpki-eku 1}

--  client of Terminal Mode
bpki-eku-clientTM OBJECT IDENTIFIER ::= {bpki-eku 2}

-- attributes
bpki-at OBJECT IDENTIFIER ::= {bpki 4}

-- certificate validity period
bpki-at-certificateValidity OBJECT IDENTIFIER ::= {bpki-at 1}

-- content types
bpki-ct OBJECT IDENTIFIER ::= {bpki 5}

bpki-ct-enroll1-req OBJECT IDENTIFIER ::= {bpki-ct 1}
bpki-ct-enroll2-req OBJECT IDENTIFIER ::= {bpki-ct 2}
bpki-ct-reenroll-req OBJECT IDENTIFIER ::= {bpki-ct 3}
bpki-ct-spawn-req OBJECT IDENTIFIER ::= {bpki-ct 4}
bpki-ct-setpwd-req OBJECT IDENTIFIER ::= {bpki-ct 5}
bpki-ct-revoke-req OBJECT IDENTIFIER ::= {bpki-ct 6}
bpki-ct-resp OBJECT IDENTIFIER ::= {bpki-ct 7}

BPKIRevokeReq ::= SEQUENCE {
  issuer          Name,
  serialNumber    INTEGER,
  revokePwd       UTF8String,
  reasonCode      CRLReason,
  invalidityDate  GeneralizedTime OPTIONAL,
  comment         UTF8String OPTIONAL }

BPKIResp ::= SEQUENCE {
  statusInfo  PKIStatusInfo,
  requestId   OCTET STRING(SIZE(32)),
  nonce       OCTET STRING(SIZE(8)) OPTIONAL }

BPKIRetrieveReq ::= SEQUENCE {
  requestId   OCTET STRING(SIZE(32)),
```

```
    nonce       OCTET STRING(SIZE(8)) }

  PrivateKeyInfo ::= SEQUENCE {
    version                  INTEGER(0),
    keyAlgorithm             CHOICE {
      bignPrivkeyAlgorithm   BignAlgorithmIdentifier,
      belsSharekeyAlgorithm  BelsAlgorithmIdentifier },
    key                      OCTET STRING }

  BignAlgorithmIdentifier ::= SEQUENCE {
    algorithm  OBJECT IDENTIFIER(bign-pubkey),
    params     OBJECT IDENTIFIER(bign-curve256v1 | bign-curve384v1 |
                                 bign-curve512v1) }

  BelsAlgorithmIdentifier ::= SEQUENCE {
    algorithm  OBJECT IDENTIFIER(bels-share),
    params     OBJECT IDENTIFIER(bels-m0128v1 | bels-m0192v1 | bels-m0256v1) }

  EncryptedPrivateKeyInfo ::= SEQUENCE {
    encryptionAlgorithm  EncryptionAlgorithmIdentifier,
    encryptedData        OCTET STRING }

  EncryptionAlgorithmIdentifier ::= SEQUENCE {
    algorithm   OBJECT IDENTIFIER(id-PBES2),
    params      PBES2-params }

  PBES2-params ::= SEQUENCE {
    keyDerivationFunc PBKDF2AlgorithmIdentifier,
    encryptionScheme  BeltKeywrapAlgorithmIdentifier }

  PBKDF2AlgorithmIdentifier ::= SEQUENCE {
    algorithm   OBJECT IDENTIFIER(id-PBKDF2),
    params      PBKDF2-params }

  BeltKeywrapAlgorithmIdentifier ::= SEQUENCE {
    algorithm   OBJECT IDENTIFIER(belt-keywrap256),
    params      NULL }

  PBKDF2-params ::= SEQUENCE {
    salt            OCTET STRING(SIZE(8)),
    iterationCount  INTEGER (10000..MAX),
    prf             PrfAlgorithmIdentifier }

  PrfAlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER(hmac-hbelt),
    params    NULL }
END
```

# Библиография


[1] Mockapetris, P., Domain names – concepts and facilities. Request for Comments: 1034, 1987
(Доменные имена – концепции и средства)

[2] Fielding R., Reschke J. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Request for Comments: 7230, 2014
(Протокол передачи гипертекста (HTTP/1.1): Синтаксис сообщений и маршрутизации)

[3] Berners-Lee T., Fielding R., Masinter L. Uniform Resource Identifier (URI): Generic Syntax. Request for Comments: 3986, 2005
(Унифицированный идентификатор ресурса (URI): Общий синтаксис)

[4] ITU-T Recommendation X.509 (2000) | ISO/IEC 9594-8:2001 Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks
(Информационные технологии. Взаимосвязь открытых систем. Директория: Инфраструктуры сертификатов открытых ключей и атрибутных сертификатов)

[5] ITU-T Recommendation X.520 (2001) | ISO/IEC 9594-6:2001, Information Technology — Open Systems Interconnection — The Directory: Selected Attribute Types, 2001
(Информационные технологии. Взаимосвязь открытых систем. Директория: Выбранные типы атрибутов)

[6] ISO 3166-1:2006 Codes for the representation of names of countries and their subdivisions — Part 1: Country codes
(Коды для представления названий стран и их областей. Часть 1: Коды стран)

[7] Eastlake 3rd D., Jones P. US Secure Hash Algorithm 1 (SHA1). Request for Comments: 3174, 2001
(Алгоритм хэширования SHA1)

[8] Nystrom M., Kaliski B. Public-Key Cryptography Standards (PKCS)#9: Selected Object Classes and Attribute Types Version 2.0. Request for Comments: 2985, 2000
(Стандарты криптографии с открытым ключом (PKCS)#9: Некоторые классы объектов и типы атрибутов версии 2.0)

[9] ISO/IEC 10646:2012 Information technology — Universal Coded Character Set (UCS)
(Информационные технологии. Универсальный набор кодированных символов (UCS))

[10] Kaliski B. Public-Key Cryptography Standards (PKCS)#5: Password-Based Cryptography Specification Version 2.0. Request for Comments: 2898, 2000
(Стандарты криптографии с открытым ключом (PKCS)#5: Криптография на основе пароля версии 2.0)

[11] Kaliski B. Public-Key Cryptography Standards (PKCS)#8: Private-Key Information Syntax Specification Version 1.2. Request for Comments: 5208, 2008
(Стандарты криптографии с открытым ключом (PKCS)#8: Синтаксис описания личного ключа версии 1.2)