RFC 5280
с дополнениями из RFC 6818
категория: стандарт

май 2008 года
январь 2013 года

ОПИСАНИЕ СЕРТИФИКАТОВ И СПИСКОВ ОТОЗВАННЫХ СЕРТИФИКАТОВ ДЛЯ X.509/PKI-ИНФРАСТРУКТУРЫ ИНТЕРНЕТ-СЕТИ

I ВВЕДЕНИЕ

Данный стандарт является частью семейства стандартов X.509/PKI-инфраструктуры в рамках Интернет-сети.

В этом стандарте представлены формат и семантика (конструкция) сертификатов и списков отозванных сертификатов (COC, *certificate revocation lists — CRL*) для PKI-инфраструктуры в Интернет-сети (Интернет/PKI-инфраструктуры). В нём также рассматриваются процедуры для обработки маршрутов сертификации (MC), а в приложениях приведены ASN.1-модули всех структур данных, которые рассматриваются или упоминаются.

II ТРЕБОВАНИЯ И ПРЕДПОЛОЖЕНИЯ

Цель данного стандарта — изложить описание X.509-сертификатов, используемых в прикладных системах Интернет-сообщества, и предоставить информацию для тех корпоративных систем, владельцы которых желают использовать X.509-технологию. К таким прикладным системам относятся всемирная W^3-сеть, электронная почта, системы аутентификации пользователей и IPsec-архитектура. В целях преодоления некоторых препятствий, связанных с использованием X.509-сертификатов, данный стандарт включает рекомендации по дальнейшему развитию систем обеспечения сертификатами, разработке прикладных программных средств и обеспечению функциональной совместимости, определяемой политикой. Некоторым организациям может понадобиться дополнить, или может быть заменить данное описание с целью удовлетворения требованиям специализированных прикладных систем, функционирующим в границах определённых сетевых сегментов или областей, в которых установлены дополнительные требования, затрагивающие авторизацию, обеспечение га-

рантий или конкретные функциональные аспекты. Тем не менее, при использовании основных прикладных систем, общее представление часто используемых атрибутов таково, что разработчики прикладных систем могут почерпнуть в данном описании всю необходимую информацию без учёта специфики организации, выпустившей соответствующий сертификат (СЕРТ) или СОС.

Пользователь СЕРТ, прежде чем начать пользоваться услугами служб аутентификации или обеспечения неотказуемости, функционирующих на основе открытого ключа, содержащегося в соответствующем СЕРТ, должен оценить (проанализировать) политику сертификации (ПЛС), установленную уполномоченным органом сертификации (*certification authority*, удостоверяющий центр — УЦ). В этой связи, данный стандарт не устанавливает каких-либо юридически обязательных правил или ограничений.

Так как могут появиться средства дополнительной авторизации и обеспечения атрибутами, например, атрибутные сертификаты, то было бы целесообразным ограничивать число атрибутов аутентификации, которые содержаться в СЕРТ. Существуют и другие средства обеспечения атрибутной информацией, которые могут предоставить наиболее предпочтительные методы доставки атрибутов аутентификации.

§2.1 Связь и топология

Владельцы СЕРТ будут их использовать в широком диапазоне прикладных систем и процессов без «оглядки» на структуру и топологию соединений и качество связи, особенно это касается пользователей защищённой электронной почты. Это описание даёт пользователям возможность «не обращать своего внимания» на полосу пропускания (скорость передачи данных), режим реального времени или высокий коэффициент готовности соединения (линии/канала связи). Более того, данный стандарт допускает наличие в линиях/каналах связи сетевых экранов (*firewall*) или иных сетевых средств фильтрации трафика.

Данное описание не рассматривает применение Системы Единого каталога (СЕК) на основе, либо Рекомендации ITU-T X.500, либо протокола упрощён-

ного доступа к СЕК-серверу (*Lightweight Directory Access Protocol* — LDAP, RFC-4510). Стандарт не запрещает использование СЕК-системы на основе Рекомендации ITU-T X.500 или LDAP-протокола. Вместе с тем, можно использовать и другие средства для распространения СЕРТ и СОС.

§2.2 Критерий приемлемости

Целью Интернет/PKI-инфраструктуры является удовлетворение потребностей в сетевой реализации детерминированных функций автоматизированных систем (служб) идентификации, аутентификации, управления доступом и авторизации. Основой для функционирования таких систем (служб) является, в том числе, и совокупность атрибутов, содержащихся в СЕРТ, а также вспомогательная управляющая информация в СЕРТ, например, информация о ПЛС и ограничения на МС.

§2.3 Перспективы для пользователей

Пользователями Интернет/PKI-инфраструктуры являются физические лица и процессы, которые используют клиентское ПО и указаны в СЕРТ в качестве их владельцев, имеющих конкретное имя (название). К пользователям относятся отправители и получатели электронных почтовых сообщений, W^3-клиенты, W^3-серверы и администратор по обеспечению криптоключами для IP-sec-архитектуры, реализованной в ПО маршрутизатора. Данное описание устанавливает ограничения на программно-аппаратные платформы, используемые такими пользователями, и ограничения, вызванные мошенничеством и отсутствием внимания со стороны самих пользователей. Это проявляется в ответственности пользователя за проведение минимально необходимых настроек (например, ключи доверенного УЦ, правила), в ограничениях на использование программно-аппаратных платформ, указанных в явном виде в самом СЕРТ, в ограничениях МС, которые защитят пользователя от многих вредоносных действий, и обеспечат его услугами прикладных систем, которые корректно реализуют функции подтверждения подлинности в автоматическом режиме.

§2.4 Перспективы для администраторов

Как и случае с пользователями, описание Интернет/PKI-инфраструктуры структурировано так, чтобы оказать помощь лицам, которые работают в органах (организациях) по сертификации. Администраторам предоставлен неограниченный выбор, который увеличивает вероятность того, что мало заметная ошибка администратора УЦ повлечёт за собой откровенную и полномасштабную дискредитацию. К тому же, неограниченный выбор существенно усложняет ПО, которое обрабатывает сформированные УЦ СЕРТ и подтверждает их подлинность.

III ОБЗОР МОДЕЛИ X.509/PKI-ИНФРАСТРУКТУРЫ

Далее рассматривается иерархическая модель PKI-инфраструктуры, представленная в Рекомендации ITU-T X.509. Компонентами этой модели являются (рис. 1):

- конечный PKI-пользователь — пользователь PKI-сертификатов и/или система, обслуживающая конечных пользователей, который(ая) является владельцем СЕРТ;
- УЦ — удостоверяющий центр (орган сертификации);
- РЦ — центр регистрации (т.е. дополнительная система, которой УЦ делегировал некоторые функции обеспечения СЕРТ);
- издатель СОС — система, которая формирует и подписывает СОС;
- репозитарий — система или совокупность распределённых систем, которая хранит СЕРТ и СОС, а также служит средством распространения этих СЕРТ и СОС среди конечных PKI-пользователей.

УЦ несут ответственность за указание состояния аннулирования СЕРТ, которые они выпустили. Информация о состоянии отзыва (аннулирования) может распространяться с помощью интерактивного протокола определения состояния сертификата (*Online Certificate Status Protocol* — OCSP, RFC-6960), списков отозванных СЕРТ или иным способом. В общем, когда информация о

состоянии отзыва (аннулирования) распространяется с использованием СОС, тогда УЦ также является издателем СОС. Тем не менее, УЦ может делегировать свои полномочия по изданию СОС другому субъекту.

(Замечание. УЦ, издающий атрибутные СЕРТ (СЕРТ|АТ), также может делегировать свои полномочия по опубликованию СОС издателю СОС.)

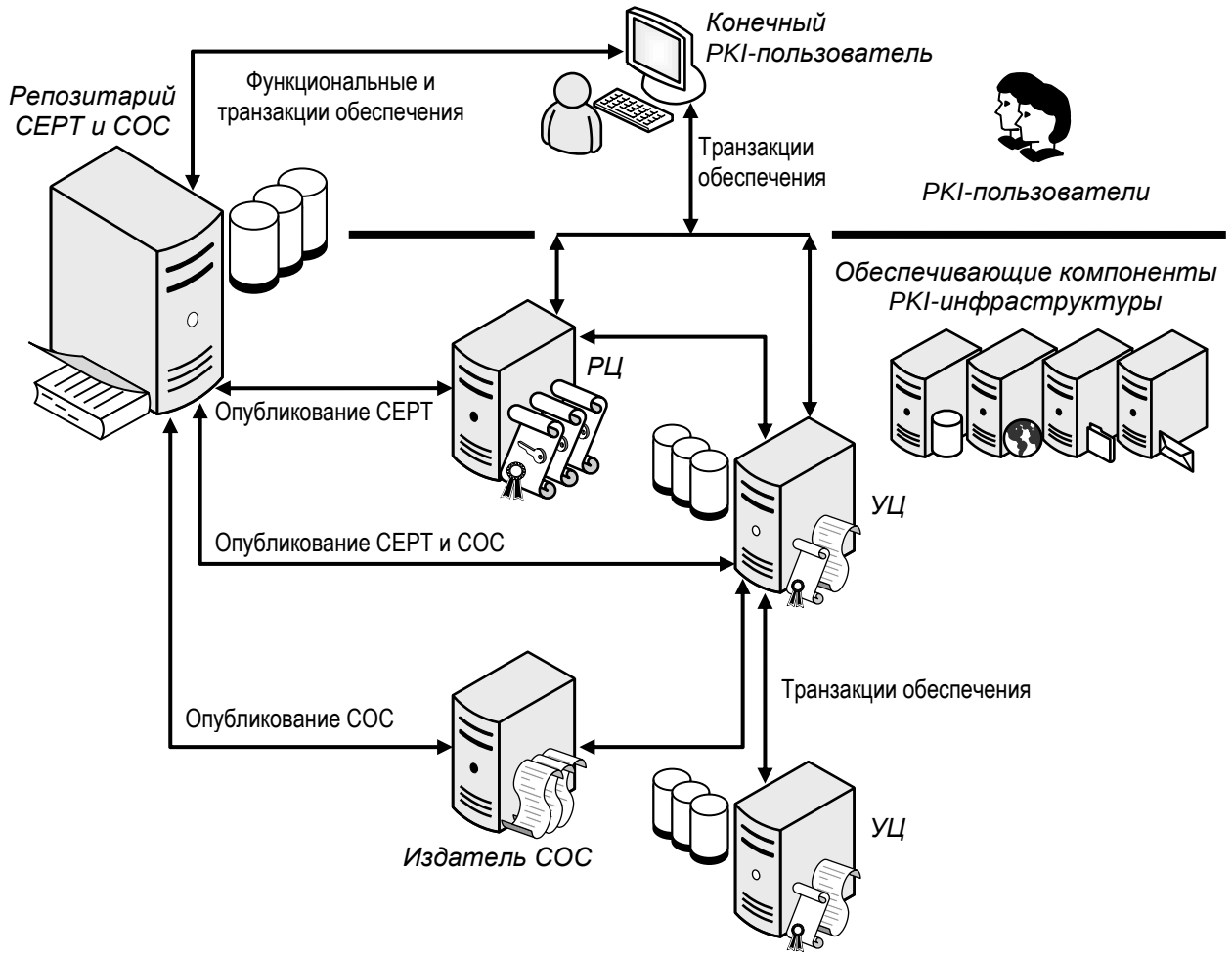

Рис.1. Компоненты PKI-инфраструктуры

§3.1 X.509-сертификат третьей версии

Пользователи открытого ключа требуют гарантий того, что соответствующий закрытый ключ принадлежит истинному удалённому субъекту (физическому лицу или системе), чтобы в дальнейшем можно было использовать этот открытый ключ в криптографических процедурах зашифрования или проверки электронной цифровой подписи (ЭЦП). Такие гарантии предоставляются путём

использования СЕРТ открытых ключей (СЕРТ|ОК), т.е. структурами данных (определённого формата), которые «привязывают» значения открытых ключей к субъектам. Привязка подтверждается (защищается) с помощью ЭЦП, проставляемой доверенным УЦ на каждом СЕРТ|ОК. УЦ может обосновывать это утверждение техническими средствами (так называемая защита собственности посредством протокола, реализующего запросно/ответный способ информационного обмена), путём предъявления закрытого ключа или обосновывать утверждение с помощью самого субъекта (владельца СЕРТ|ОК). СЕРТ|ОК имеет ограниченный срок действия, который указывается в подписанном содержании самого сертификата. Так как подпись сертификата и время его действия могут быть проверены клиентом, использующим этот СЕРТ|ОК, независимо, то сертификаты могут распространяться по не надёжным соединениям (линиям/каналам связи) и через сѐрверы прикладных систем, а также могут храниться в незащищённых базах данных (хранилищах) прикладных систем, использующих сертификаты.

Рекомендация ITU-T X.509 (бывший CCITT X.509) или стандарт ISO/IEC 9594-8, которые были впервые опубликованы в 1988 году в качестве составной части совокупности Рекомендаций ITU-T X.500, описывающих Службу единого каталога, устанавливают стандартный формат СЕРТ. В 1988 году это была первая версия (v1) формата. В 1993 году Рекомендации ITU-T X.500 прошли соответствующую ревизию, в результате которой балы принята вторая версия формата (v2), включавшего два дополнительных поля.

Ряд Интернет-стандартов (*Request for Comment* — RFC), опубликованных в 1993 году, описывали частную усовершенствованную службу электронной Интернет-почты (*Internet Privacy Enhanced Mail* — PEM), включающую PKI-инфраструктуру, которая базировалась на сертификатах Рекомендации ITU-T X.509v1 (RFC-1422). Опыт, накопленный при развёртывании системы в соответствие с RFC-1422, ясно дал понять, что форматы сертификатов первой и второй версий во многих отношениях были неприемлемы. Очень важным оказалось то, что для доставки данных, которые были связаны с проектными осо-

бенностями PEM-системы, необходимы дополнительные поля, а последующие годы внедрения и эксплуатации этой системы подтвердили такую необходимость. В ответ на эти возникшие требования, ISO/IEC, ITU-T и ANSI X9 усовершенствовали и приняли новую третью версию формата сертификата. Третья версия формата сертификата (v3) расширяет формат второй версии путём введения дополнительных субполей в поле «Расширения». Соответствующие типы субполей в поле «Расширения» могли быть описаны в стандартах или могли быть определены и зарегистрированы любой организацией или сообществом. В июне 1996 года стандартизация основной третьей версии формата сертификата была завершена.

ISO/IEC, ITU-T и ANSI X9 также доработали стандартное поле «Расширения» с целью его использования в третьей версии этого поля. Новые субполя в поле «Расширения» могут доставлять различную информацию, например, дополнительную идентификационную информацию субъекта, атрибутную информацию о ключе, информацию о политике сертификации и ограничения на МС.

Тем не менее, стандартное поле «Расширения», установленное стандартами ISO/IEC, ITU-T и ANSI X9, обеспечивает широкий диапазон его применения в интересах различных прикладных систем. С целью обеспечения функциональной совместимости прикладных систем, использующих X.509v3-сертификаты, в рамках Интернет-сети, необходимо определить структуру субполей поля «Расширения» в таких X.509v3-сертификатах, которая будет приемлема для её использования в Интернет-сети. Одной из целей данного стандарта дать описание структуры и формата CERT и COC для их использования в W^3-сети, системе электронной почты и прикладных системах, основанных на IPsec-архитектуре. Прикладные системы, предъявляющие дополнительные требования, могут быть построены на основе данного описания или могут его заменить.

§3.2 Маршруты сертификации и доверие

Клиент службы обеспечения безопасности, для которого необходимо знать открытый ключ, как правило, нуждается в получении и проверке подлинности СЕРТ, содержащего требуемый ключ. Если пользователь открытого ключа уже не обладает надёжной копией открытого ключа УЦ, подписавшего сертификат, наименование этого УЦ и дополнительной информацией (например, периодом действия или ограничения на именование), то может понадобиться дополнительный сертификат для получения такого открытого ключа. Как правило, может понадобиться цепочка нескольких связанных сертификатов, включающая сертификат владельца открытого ключа (конечного субъекта), подписанный одним УЦ, и один или несколько (либо ни одного) сертификатов УЦ, подписанных другими УЦ. Такие цепочки сертификатов, именуемые МС, чрезвычайно необходимы пользователю открытого ключа, так как именно с него начинается ограниченное количество (последовательность) открытых ключей, гарантированных УЦ.

Существует несколько способов упорядочения УЦ, которое необходимо пользователям открытых ключей с целью поддержки их при поиске МС. Для PEM-системы стандарт RFC-1422 определил жёсткую иерархическую структуру УЦ. В PEM-системе существуют следующие три типа УЦ:

a. Центр регистрации политики в Интернет-сети (*Internet Policy Registration Authority* — IPRA). Этот центр, функционирующий под эгидой Интернет-сообщества, выступает в роли *корневого УЦ первого уровня* в иерархической структуре сертификации PEM-системы. Он выпускает СЕРТ только для УЦ следующего уровня (УЦ, формирующие политики сертификации). Все МС начинаются с IPRA-центра;

b. УЦ, формирующие политики сертификации (*Policy Certification Authorities* — PCA). PCA-центры располагаются на *втором уровне* иерархии, а каждый PCA-центр сертифицируется IPRA-центром. PCA-центр разрабатывает и опубликовывает положения своей политики, связанной с сертификацией пользователей или подчинённых УЦ. Специализированные PCA-центры предназначены для удовлетворения различных запросов пользователей. Например, один

PCA-центр (организационный PCA-центр) может обеспечивать потребности коммерческих организаций в услугах общей электронной почты, а другой PCA-центр (высоко надёжный PCA-центр) может иметь более жёсткую политику, предназначенную для юридически обязательных требований по использованию ЭЦП;

c. УЦ (*Certification Authorities* — CA). УЦ образуют третий уровень иерархии и могут располагаться на более низких уровнях. УЦ третьего уровня сертифицируются PCA-центрами. УЦ представляют, например, соответствующие организации, подразделения таких организаций (например, департаменты, группы, отделы) или соответствующие географические области.

Стандарт RFC-1422 содержит правило именования подчинённых субъектов, которое требует, чтобы УЦ мог выпускать сертификаты только для тех субъектов, чьи имена являются производными (подчинёнными) по отношению к наименованию самого этого УЦ. Надёжность, касающаяся МС в PEM-системе, «вытекает» из наименования PCA-центра. Такое правило именования подчинённых субъектов гарантирует, что УЦ, расположенные в иерархии ниже PCA-центров, корректно разделены на группы подчинённых субъектов, которые могут сертифицироваться этими PCA-центрами (например, УЦ организации может сертифицировать только те субъекты, которые входят в структуру дерева именования субъектов этой организации). Системы сертификации пользователей могут автоматически проверять выполнение правила именования подчинённых субъектов.

Стандарт RFC-1422 использует формат сертификата, рекомендованный ITU-T X.509v1. Ограничения Рекомендации ITU-T X.509v1 требуют установления нескольких ограничений на структуру информации о соответствии с предшествующими политиками сертификации или ограничений применимости сертификатов. К таким ограничениям относятся:

a) строгая иерархия «сверху-вниз», т.е. все МС начинаются с IPRA-центра;

b) правило именования подчинённых субъектов ограничивает имена подчинённых субъектов УЦ;

c) использование концепции PCA-центра, что требует знания конкретных PCA-центров с целью построения логической цепочки проверки, размещаемой в сертификате. Знание конкретных PCA-центров также необходимо при определении, а могла ли цепочка быть приемлемой.

С принятием третьей версии Рекомендации ITU-T X.509v3 большинство требований, выдвинутых стандартом RFC-1422, могут быть удовлетворены на основе использования дополнительных субполей поля «*Расширения*» в CERT и без необходимости ограничения уже используемых структур УЦ. Соответственно субполя поля «Расширения» в CERT, связанные с политиками сертификации, исключают необходимость применения PCA-центров, а субполя, связанные с ограничениями, исключают необходимость использования правил именования подчинённых субъектов. В результате, данный стандарт предлагает более гибкую архитектуру, которая включает:

a) МС начинаются с открытого ключа УЦ, расположенного в собственном сетевом сегменте и обслуживающего пользователей в рамках этого сегмента, или с открытого ключа корневого УЦ в иерархии. Начало маршрута с открытого ключа УЦ, расположенного в собственном сетевом сегменте и обслуживающего пользователей в рамках этого сегмента, имеет ряд существенных преимуществ. В некоторых прикладных системах доверяют больше локальному сегменту;

b) ограничения на процедуры именования могут быть наложены путём явного включения в CERT дополнительного субполя в поле «*Расширения*», устанавливающего такие ограничения, но последние не востребованы;

c) дополнительные субполя в поле «*Расширения*», определяющие политику сертификации и отображение политики делают ненужной концепцию PCA-центра, которая допускает высокий уровень автоматизации процедуры. Прикладной процесс (система) может определить, является ли МС допустимым, основываясь лишь только на содержании сертификатов, а не на априорном знании PCA-центров. Эта также допускает автоматизированную обработку МС.

Рекомендация ITU-T X.509v3 также вводит дополнительное субполе в поле «*Расширения*» CERT, идентифицирующее владельца сертификата, либо как действующий УЦ, либо как дееспособный конечный пользователь, снижая, таким образом, зависимость от вспомогательной информации, передача которой востребована в PEM-системе.

Данный документ затрагивает два класса CERT:

- сертификаты УЦ (*CA certificates*). В дальнейшем сертификаты УЦ могут подразделяться на три класса:
  - кросс-сертификаты (*cross-certificates*). Кросс-сертификаты представляют собой сертификаты УЦ, в которых издатель и владелец являются разными субъектами. Кросс-сертификаты отображают доверенную взаимосвязи между двумя УЦ;
  - само-изданные сертификаты (*self-issued certificates*). Само-изданные сертификаты представляют собой сертификаты УЦ, в которых издатель и владелец являются одним и тем же субъектом. Само-изданные сертификаты выпускаются для опубликования изменений, которые затрагивают политику сертификации или функциональные процедуры;
  - само-подписанные сертификаты (*self-signed certificates*). Само-подписанные сертификаты представляют собой само-изданные сертификаты, ЭЦП в которых может быть проверена с помощью открытого ключа, «привязанному» к этому сертификату. Само-подписанные сертификаты применяются при доставке открытого ключа для использования его в начале МС;
- сертификаты конечных пользователей (*end entity certificates*). Сертификаты конечных пользователей выпускаются для субъектов, которые не уполномочены для издания сертификатов.

Следуя параграфу §3.4.61 Рекомендации ITU-T X.509 (опубликована в ноябре 2008 года), необходимо отметить, что само-изданные и само-подписанные сертификаты, издаваемые субъектами, которые не являются УЦ, в данном стандарте не рассматриваются. Следовательно, например, W^3-сервер или его

клиент могут формировать само-подписанные сертификаты с целью собственной идентификации. В этой связи, такие сертификаты и процедурные вопросы, касающиеся их использования при подтверждении параметров подлинности взаимодействующих сторон, в данном стандарте также не рассматриваются.

§3.3 Аннулирование (отзыв)

Когда СЕРТ издан, то предполагается, что он будет использоваться в течение всего своего периода действия. Однако могут произойти различные события, в результате которых СЕРТ станет недействительным ещё до окончания своего периода действия. К таким событиям относятся изменение имени, изменение взаимосвязи между субъектом и УЦ (например, работник завершает трудовую деятельность в организации), компрометация или подозрение на компрометацию соответствующего закрытого ключа. В случае возникновении таких событий УЦ необходимо аннулировать СЕРТ.

Рекомендация ITU-T X.509v3 описывает всего один метод отзыва СЕРТ. Этот метод предписывает каждому УЦ периодически издавать структурированный информационный объект, подписанный этим УЦ, который называется списком отозванных (аннулированных) сертификатов (СОС, *certificate revocation list* — CRL). СОС представляет собой список, содержащий метку времени и идентифицирующий отозванные сертификаты, и который подписывается УЦ или издателем СОС, а также помещается в открытый репозитарий для публичного доступа. Каждый отозванный и помещённый в СОС СЕРТ идентифицируется с помощью своего последовательного номера. Когда прикладная система, предусматривающая применение СЕРТ, использует последний (например, для проверки ЭЦП удалённого пользователя), то такая система не только проверяет подпись и срок действия СЕРТ, но также запрашивает соответствующий последний (*suitably recent*) СОС и проверяет, последовательный номер СЕРТ не содержится в СОС. Смысл понятия «соответствующий последний» может варьироваться в соответствие с локальной политикой сертификации, тем не менее, как правило, это означает самый «свежий» изданный СОС. СОС обновля-

ется на регулярной основе (например, каждый час, ежедневно или еженедельно). В СОС добавляется запись, как часть последующего обновления, сразу после уведомления об аннулировании. Запись никогда не должна удаляться из СОС до тех пор, пока она не появиться в одном из регулярно публикуемых СОС, изданном после окончания срока действия аннулированного сертификата.

Преимуществом этого метода отзыва является то, что СОС могут распространяться с помощью точно таких же средств как и сами СЕРТ, а именно по не надёжным соединениям (линиям/каналам связи) и через сѐрверы прикладных систем.

При использовании не надёжных соединений (линий/каналов связи) и сѐрверов прикладных систем, для метода аннулирования на основе СОС существует одно ограничение, которое заключается в том, что регулярность (частота) аннулирования ограничивается периодом издания СОС. Например, если об аннулировании СЕРТ объявлено прямо сейчас, то прикладным системам, предусматривающим применение СЕРТ, не будет сообщено о таком факте аннулирования до тех пор, пока все изданные на настоящий момент СОС не будут в плановом порядке обновлены (а это может быть ежечасно, ежедневно или еженедельно, в зависимости от частоты переиздания СОС).

Как и в случае с форматом сертификата Рекомендации ITU-T X.509v3, с целью обеспечения функциональной совместимости внедряемых несколькими вендорами прикладных систем, необходимо описать формат СОС Рекомендации ITU-T X.509v2 при его использовании в Интернет-сети. Одной из целей этого документа является описание такого формата. Однако, данное описание не требует выпуска СОС. Форматы сообщений и протоколы, обеспечивающие интерактивное оповещение об аннулировании, определены в других стандартах в интересах глобальной PKI-инфраструктуры. Интерактивные методы оповещения об аннулировании могут быть приемлемы в некоторых прикладных сферах в качестве альтернативы СОС Рекомендации ITU-T X.509. Интерактивная проверка отзыва может существенно снизить время ожидания между сообщением об аннулировании и доведением информации до взаимодействующих

сторон. Так как УЦ воспринимает сообщение об аннулировании как подлинное и действующее, любое обращение к интерактивной службе по проверке подлинности сертификата корректно воспримет все изменения, вызванные аннулированием. Тем не менее, такие методы обуславливают появление новых требований по обеспечению информационной безопасности. Т.е. средству проверки подлинности СЕРТ необходима надёжная интерактивная служба проверки подлинности, за исключением репозитария, последнему не надо быть доверенным.

§3.4 Функциональные протоколы

Функциональные протоколы необходимы для доставки СЕРТ и СОС (или информации о их состоянии) прикладным пользовательским системам, предусматривающим применение СЕРТ. Необходимы и дополнительные меры по обеспечению всего многообразия средств доставки СЕРТ и СОС, включая процедуры распространения, основанные на LDAP-, HTTP- и FTP-протоколах, а также на Рекомендации ITU-T X.509. Функциональные протоколы, реализующие такие функции представлены в других стандартах в интересах глобальной PKI-инфраструктуры. В таких стандартах могут быть представлены описания форматов сообщений и процедуры для поддержки всех указанных выше функциональных прикладных систем, включая описания или ссылки на соответствующие типы MIME-сообщений (*Multipurpose Internet Mail Extensions* — RFC-2045...2049).

§3.5 Протоколы обеспечения

Протоколы обеспечения (или обеспечивающие протоколы) необходимы для обеспечения интерактивного взаимодействия между PKI-пользователем и обеспечивающими субъектами. Например, обеспечивающий протокол может использоваться при взаимодействии УЦ и клиентской системой, с которой связаны пара ключей, либо — между двумя УЦ, которые кросс-сертифицированы друг с другом. К функциям, которые потенциально необходимы и которые реализуются с помощью обеспечивающих протоколов, относятся:

a) регистрация: это процесс, с помощью которого пользователь первым оповещает УЦ о себе (напрямую или через РЦ), ещё до того, как УЦ выпустит сертификат или сертификаты для этого пользователя;

b) инициализация: прежде чем клиентская система сможет функционировать в защищённом режиме, необходимо инсталлировать ключевую информацию, которая соответствующим образом связана с криптоключами, хранящимися в другом месте инфраструктуры. Например, клиенту необходимо начать работу с открытым ключом и другой достоверной информацией надёжного УЦ (или нескольких УЦ) в защищённом режиме, которая будет использоваться при проверке подлинности МС;

Более того, как правило, клиенту необходимо инициализироваться для работы со своими собственными ключами (парами ключей).

c) сертификация: это процесс, в котором УЦ выпускает СЕРТ открытого ключа пользователя, направляет этот СЕРТ|ОК клиентскую систему, обслуживающую пользователя, и/или передаёт этот СЕРТ|ОК в репозиторий;

d) восстановление пары ключей: это дополнительная (не обязательная) функция. Информация о клиентском ключе пользователя (например, закрытый ключ пользователя, используемый для шифрования) может быть заархивирована УЦ или системой архивирования ключей. Если пользователю необходимо восстановить эту заархивированную информацию о ключе (например, в результате забывания пароля или потери файла с последовательностью ключей), то для обеспечения такого восстановления может понадобиться интерактивный протокол информационного обмена;

e) обновление пары ключей: все пары ключей необходимо регулярно обновлять (т.е. заменять на новую пару ключей), а также необходимо выпускать новые сертификаты;

f) запрос аннулирования: правомочная персона оповещает УЦ о нештатной ситуации, требующей аннулирования сертификата;

g) кросс-сертификация: два УЦ обмениваются информацией, используемой при формировании кросс-сертификата. Кросс-сертификат представляет собой сертификат, выпущенный одним УЦ для другого УЦ, и который содержит ключ подписи УЦ, используемый при выпуске сертификатов.

(Примечание. Интерактивные протоколы являются не единственным способом реализации рассмотренных выше функций. Для реализации всех функций существуют автономные методы, которые обеспечивают получение аналогичных результатов, а данный стандарт не предписывает использование интерактивных протоколов. Например, если используются съёмные носители информации (СНИ, *hardware tokens*), то многие функции могут быть реализованы путём физической доставки таких СНИ. Более того, некоторые из функций могут быть включены в один протокол информационного обмена. Соответственно, две или более функций регистрации, инициализации и сертификации могут быть объединены в одном протоколе информационного обмена.)

Ряд стандартов в интересах глобальной PKI-инфраструктуры определяют набор стандартных форматов сообщений, предназначенных для реализации указанный выше функций. Кроме того, в этих стандартах представлены протоколы, предназначенные для доставки таких сообщений в различные прикладные системы (например, протоколы доставки сообщений электронной почты, файлов и гипертекстовых сообщений).

IV ОПИСАНИЕ СЕРТИФИКАТА И ПОЛЯ «РАСШИРЕНИЯ» В СЕРТИФИКАТЕ

В данном разделе представлено описание CEPT|ОК, которые обеспечивают функциональную совместимость и своё многократное использование в рамках PKI-инфраструктуры. Далее представлен формат CEPT|ОК/X.509v3 и поля «*Расширения*» в сертификате. Стандарты ISO/IEC и ITU-T используют версию ASN.1-кодирования 1997 года, в данном стандарте используется версия ASN.1-кодирования 1988 года. Но при этом стандарты сертификатов и полей «*Расширения*» эквивалентны. Кроме того, далее описываются дополнительные

субполя поля «*Расширения*», предназначенные для частного применения, которые востребованы в рамках X.509/PKI-инфраструктуры в Интернет-сети.

Сертификаты могут использоваться во многих прикладных системах и областях, обеспечивая достижение поставленных целей, связанных с решением проблемы функциональной совместимости, а также удовлетворение требованиям по функциональности и гарантированности. Цель данного документа сформировать, с одной стороны, общую основу для функционирования универсальных прикладных систем, которые требуют широкомасштабной функциональной совместимости, а с другой, — ограниченные требования специального назначения. Соответственно, основной акцент будет сделан на обеспечении использования СЕРТ|ОК/X.509v3 наиболее востребованных прикладных систем Интернет-сети: электронная почта, IPsec-архитектура и W^3-сеть.

§4.1 Основные поля сертификата

Основной синтаксис СЕРТ|ОК/X.509v3 представлен ниже. При вычислении подписи, данные, которые должны быть подписаны, кодируются с использованием особых правил кодирования (*distinguished encoding rules* — DER) ASN.1-стандарта (Рекомендация ITU-T X.690). DER/ASN.1-кодирование включает дескриптор, размер и систему кодирования значения каждого элемента.

```
Certificate ::= SEQUENCE {
tbsCertificate                  TBSCertificate,
signatureAlgorithm              AlgorithmIdentifier,
signatureValue                  BIT STRING }

TBSCertificate ::= SEQUENCE {
version                         [0] Указывается в явном виде, в режиме «по умолчанию»
                                -- версия v1,
serialNumber                    CertificateSerialNumber,
signature                       AlgorithmIdentifier,
issuer                          Name,
validity                        Validity,
subject                         Name,
subjectPublicKeyInfo            SubjectPublicKeyInfo,
issuerUniqueID                  [1] Не обязательное поле «Уникальный идентификатор
                                -- издателя», может указываться, может не указываться
                                -- если представлено, то версия должна быть v2 или v3
subjectUniqueID                 [2] Не обязательное поле «Уникальный идентификатор
                                -- владельца», может указываться, может не указываться
                                -- если представлено, то версия должна быть v2 или v3
extensions                      [3] Указывается в явном виде, как дополнительное поле,
```

```
                                -- если представлено, то версия должна быть v3
}

Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER

Validity ::= SEQUENCE {
NotBefore                       Time,
NotAfter                        Time }

Time ::= CHOICE {
utcTime                         UTCTime,
generalTime                     GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
Algorithm                       AlgorithmIdentifier,
SubjectPublicKey                BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
extnID                          OBJECT IDENTIFIER,
critical                        BOOLEAN DEFAULT FALSE,
extnValue                       OCTET STRING
                                -- включает DER-кодировку ASN.1-стандарта,
                                -- соответствует типу поля «Расширения», определяемого
                                -- идентификатором «extnID»
}
```

Далее описываются структура и поля СЕРТ|ОК/X.509v3 для его использования в Интернет-сети.

§4.1.1 Поля сертификата

СЕРТ|ОК/X.509v3 представляет собой последовательность, состоящую из трёх обязательных полей.

§4.1.1.1 Поле «tbsCertificate»[°]

Это поле включает наименования субъекта (владельца) и издателя, открытый ключ, связанный с субъектом, период действия и иную необходимую информацию. Кроме того, данной поле, как правило включает поле «*Расширения*».

[°] Аббревиатура «tbs» означает «To Be Signed» (для подписи).

§4.1.1.2 Поле «signatureAlgorithm»

Поле «*signatureAlgorithm*» (алгоритм вычисления подписи) содержит идентификатор криптоалгоритма, используемого УЦ при формировании подписи данного СЕРТ. Стандарты RFC-3279, RFC-4055 и RFC-4491 перечисляют применяемые алгоритмы вычисления подписи. Однако, могут использоваться и другие алгоритмы.

Структура идентификатора алгоритма имеет следующее ASN.1-кодирование:

```
AlgorithmIdentifier  ::=  SEQUENCE  {
algorithm                       OBJECT IDENTIFIER,
parameters                      ANY DEFINED BY algorithm OPTIONAL  }
```

Идентификатор алгоритма используется для определения криптографического алгоритма. Субполе «*OBJECT IDENTIFIER*» определяет алгоритм (например, DSA с SHA-1). Содержание субполя «*(optional) Parameters*» (дополнительные параметры) может варьироваться (и по длине), в зависимости от указанного алгоритма.

Это поле должно в обязательном порядке содержать тот же идентификатор алгоритма, который содержится в субполе «*Signature*» (подпись) поля «*tbsCertificate*»

§4.1.1.3 Поле «signatureValue»

Поле «*signatureValue*» (результат вычисления (значение) подписи) содержит цифровую подпись, вычисленную по последовательности поля «*tbsCertificate*», представленного в DER/ASN.1-формате. Поле «*tbsCertificate*» в DER/ASN.1-формате используется в качестве входной последовательности функции вычисления подписи. Полученный результат (подпись) кодируется в форме битовой последовательности (BIT STRING) и размещается в поле «*signatureValue*».

С помощью формирования такой подписи УЦ подтверждает подлинность (сертифицирует) информации(ю) в поле «*tbsCertificate*». Соответственно

УЦ сертифицирует связку между данными об открытом ключе и владельце сертификата.

§4.1.2 Субполя (состав) поля «tbsCertificate»

Поле «*tbsCertificate*» содержит информацию, которая связана с владельцем (держателем) сертификата и выпустившим этот сертификат УЦ. Каждое поле «*tbsCertificate*» содержит наименования субъекта и издателя, связанный с субъектом открытый ключ, срок действия, номер версии и последовательный номер. Отдельные сертификаты в поле «*tbsCertificate*» могут включать дополнительные поля, содержащие уникальные идентификаторы (УИД).

§4.1.2.1 Версия «Version»

Это субполе содержит номер версии закодированного сертификата. Когда используется поле «*Расширения*», что наиболее вероятно, версия должна быть третий (значение «2»). Если же указанное поле не представлено, но представлен уникальный идентификатор («*UniqueIdentifier*»), то версия должна быть второй (значение «1»). Тем не менее, в таком случае версия может быть третьей. Если же представлены только основные поля, то версия должна быть первой (значение исключено из сертификата). Тем не менее, в таком случае версия может быть второй или третьей.

Внедряемые прикладные системы должны быть готовы к приёму сертификатов любых версий. Соответствующие прикладные системы должны, как минимум, распознавать сертификаты третьей версии.

Предполагается, что поколение сертификатов второй версии не будет применяться теми прикладными системами, которые основываются на данном описании (стандарте).

§4.1.2.2 Последовательный номер «Serial Number»

Последовательный номер должен быть положительным целым числом, назначаемым УЦ каждому сертификату. Оно должно быть уникальным для

каждого сертификата, выпущенного конкретным УЦ (т.е. имя издателя и последовательный номер идентифицируют уникальный сертификат).

УЦ должны строго контролировать процедуру издания СЕРТ, чтобы последовательный номер никогда не был отрицательным целым числом. Представленные выше требования по уникальности предполагают, что последовательные номера могут быть длинными целыми числами. Пользователи СЕРТ должны быть способны обрабатывать значение в субполе «*serialNumber*» длиной до 20 октетов (включительно). Следующие этому стандарту УЦ не должны использовать значения в субполе «*serialNumber*» длиной более 20 октетов.

(Примечание. Не выполняющие этот стандарт УЦ могут издавать СЕРТ с последовательными номерами, значения которых могут быть отрицательными или нулевыми. Пользователи СЕРТ должны быть соответствующим образом подготовлены для обработки таких сертификатов.)

§4.1.2.3 Подпись «Signature»

Это субполе содержит идентификатор алгоритма, используемого УЦ для подписи СЕРТ.

Это субполе должно содержать один тот же идентификатор алгоритма, как и в поле «*signatureAlgorithm*» последовательности «Сертификат». Содержание субполя «*(optional) Parameters*» (дополнительные параметры) может варьироваться (и по длине), в зависимости от указанного алгоритма. Стандарты RFC-3279, RFC-4055 и RFC-4491 перечисляют применяемые алгоритмы вычисления подписи. Однако, могут использоваться и другие алгоритмы.

§4.1.2.4 Издатель «Issuer»

Субполе «*Issuer*» идентифицирует субъекта, который подписал и выпустил сертификат. Это субполе должно содержать «не пустое» уникальное наименование. Данное субполе описано в Рекомендации ITU-T X.501 как тип «*Name*» (имя). Далее приведена структура этого субполя.

```
Name ::= CHOICE { -- на данный момент возможно толко одно --
rdnSequence                 RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::=
SET SIZE (1..MAX) OF        AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
type                        AttributeType,
value                       AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY -- DEFINED BY AttributeType

DirectoryString ::= CHOICE {
teletexString               TeletexString (SIZE (1..MAX)),
printableString             PrintableString (SIZE (1..MAX)),
universalString             UniversalString (SIZE (1..MAX)),
utf8String                  UTF8String (SIZE (1..MAX)),
bmpString                   BMPString (SIZE (1..MAX)) }
```

«*Имя*» (наименование) определяется как иерархическое имя, составленное из атрибутов, например, название страны и соответствующее значение, например, US. Тип компонента «*AttributeValue*» (значение атрибута) определяется как «*AttributeType*» (тип атрибута). В общем случае, он будет иметь вид «*DirectoryString*».

Тип данных «*DirectoryString*» определяет выбор кодировки: «*PrintableString*» (последовательность символов для печати), «*TeletexString*» (последовательность символов для телетекста), «*BMPString*» (битовая последовательность символов), «*UTF8String*» (последовательность символов в UTF8-коде) и «*UniversalString*» (универсальная последовательность символов). УЦ, придерживающиеся данного описания, должны использовать либо кодировку «*PrintableString*», либо кодировку «*UTF8String*», но с двумя исключениями. Если прежде УЦ выпускали сертификаты с полями «*Issuer*», в которых содержались атрибуты с использованием *TeletexString*-, *BMPString*- или *UniversalString*-кодировок, то УЦ может продолжать использовать эти кодировки для данных типа «*DirectoryString*» с целью сохранения предшествующей совместимости. Кроме того, новые УЦ, входящие в сегмент, в котором действующие УЦ выпускают сертификаты с полями «*Issuer*», содержащие атрибуты с использованием *Teletex-*

String-, *BMPString*- или *UniversalString*-кодировок, могут кодировать атрибуты, которыми они будут обмениваться с существующими УЦ, используя для этого те же самые кодировки, используемые этими действующими УЦ.

Как отмечалось ранее, уникальные имена (наименования) составляются из атрибутов. Данный документ не вводит ограничения на типы атрибутов, которые могут использоваться в именах. Однако, прикладные системы, следующие требованиям данного стандарта, должны быть готовы к приёму сертификатов, в которых имена издателей содержат представленный ниже набор атрибутов. Данный документ рекомендует поддерживать и дополнительные типы атрибутов.

Стандартные наборы атрибутов определены в Рекомендациях ITU-T X.500-серии. Прикладные системы, следующие требованиям данного стандарта, обязаны быть готовыми к приёму сертификатов, в которых имена издателей и владельцев сертификатов содержат следующие стандартные типы атрибутов:

- страна (*country*);
- организация (*organization*);
- структурное подразделение организации (*organizational unit*);
- определитель уникального имени (*distinguished name qualifier*);
- наименование штата (провинции) или региона (области) (*state or province name*);
- общепринятое имя (например, «*Susan Housley*», *common name*);
- последовательный номер (*serial number*).

Кроме того, прикладные системы, следующие требованиям данного стандарта, должны быть готовы к приёму сертификатов, в которых имена издателей и владельцев сертификатов содержат следующие дополнительные стандартные типы атрибутов:

- местоположение (*locality*);
- титул (звание) (*title*);
- фамилия (*surname*);
- имя (*given name*);

- инициалы (*initials*);
- псевдоним (*pseudonym*);
- определитель поколения (представитель одной из ступеней семейного генеалогического древа) (например, «*Jr.*» («*Мл.*»), «*3rd*» или «*IV*», *generation qualifier*).

Синтаксис и соответствующие идентификаторы объектов (*object identifier* — OID) для этих типов атрибутов представлены в ASN.1-модулях Приложения «А».

Дополнительно, прикладные системы, следующие требованиям данного стандарта, должны быть готовы к приёму атрибута «*domainComponent*», представленного в стандарте RFC-4519. Система именования сегментов/областей определяет иерархическую систему маркирования (именования) сетевых ресурсов (компонентов). Указанный атрибут предоставляет организациям, по их желанию, весьма удобный способ одновременного использования уникальных имён и их DNS-имён. Это не предусматривает замену компонента «*dNSName*» на субполе «*Альтернативное имя*» поля «*Расширения*». Прикладные системы не требуют преобразования таких имён в DNS-имена.

Пользователи сертификатов должны быть готовы к обработке полей, содержащих уникальное имя издателя сертификата и уникальное имя владельца сертификата, с целью построения «цепочки имён» (*name chaining*) при проверке подлинности МС. Цепочка имён проверяется путём сравнения уникального имени издателя сертификата в одном сертификате с уникальным именем владельца сертификата, указанного в сертификате УЦ. Если имена в поле «*Issuer*» и в поле «*Subject*» сертификата совпадают, в соответствие с принятыми правилами, то имеет место само-изданный сертификат.

§4.1.2.5 Период действия «Validity»

Период действия сертификата представляет собой временной интервал, в течении которого УЦ гарантирует, что он будет обновлять информацию о со-

стоянии сертификата. Это поле представляет собой последовательность двух дат (субполей):

1. дата начала периода действия («*notBefore*»);
2. дата окончания периода действия («*notAfter*»).

Обе эти даты могут кодироваться как UTCTime-время или GeneralizedTime-время.

УЦ, придерживающиеся данного стандарта, обязаны всегда кодировать даты периода действия сертификата как UTCTime-время до 2049 года включительно. А начиная с 2050 года и далее — как GeneralizedTime-время. Прикладные системы, которые также придерживаются данного стандарта, должны быть способны обрабатывать даты периода действия сертификата, закодированные как UTCTime-время или как GeneralizedTime-время.

Срок действия сертификата это период времени от даты, указанной в субполе «*notBefore*», до даты, указанной в субполе «*notAfter*», включительно.

В отдельных случаях, программно-аппаратным комплексам (ПАК) могут выдаваться сертификаты, в которых будут указаны не совсем корректные сроки действия. Например, ПАК может быть выдан сертификат, в котором номер модели и серийный номер ПАК будут привязаны к его открытому ключу. Такой сертификат предназначен для использования на протяжении всего жизненного цикла ПАК.

Для указания того, что сертификат включает некорректный срок действия, дата, указанная в субполе «*notAfter*», должна иметь кодировку GeneralizedTime и значение «*99991231235959Z*».

Когда издатель не способен обновлять (корректировать) информацию о статусе (состоянии) до тех пор, пока не наступит дата «*notAfter*» (включая случай, когда дата «*notAfter*» имеет значение «*99991231235959Z*»), тогда он обязан гарантировать, что после прекращения обновления информации о состоянии данного сертификата, для последнего не существует приемлемого МС. Это может можно обеспечить, либо на основе завершения срока действия, либо путём аннулирования (отзыва) все сертификатов УЦ, которые содержат открытый

ключ для проверки подписи сертификата, а также путём прекращения использования открытого ключа для проверки подписи сертификата, выступающей в качестве «якоря доверия» (*trust anchor*, в данном случае это последняя «скрепляющая» подпись, замыкающая цепочку (маршрут или путь) доверия или сертификации).

§*4.1.2.5.1 Время UTCTime*

Универсальный тип времени, UTCTime-время, является стандартным типом в ASN.1-коде, предназначенным для кодировки дат и времени. UTCTime-время определяет год с помощью двух цифр в нижнем регистре и время — с точность до одной минуты или секунды. UTCTime-время включает, либо значение «Z» (время по Гринвичу), либо разницу во времени.

В рамках данного стандарта, значения UTCTime-времени должны обязательно отображать время по Гринвичскому (нулевому) меридиану и включать секунды (например, время «*YYMMDDHHMMSSZ*»), и даже в тех случаях, когда число секунд равно нулю. Основанные на данном стандарте системы обязаны интерпретировать субполе «*year*» («*YY*», год) следующим образом:

a) если значение «*YY*» больше или равно 50, то год интерпретируется как «*19YY*»;

b) если значение «*YY*» меньше 50, то год интерпретируется как «*20YY*».

§*4.1.2.5.2 Время GeneralizedTime*

Обобщённый тип времени, GeneralizedTime-время, является стандартным типом в ASN.1-коде, предназначенным для изменяющегося точного представления времени. Кроме того, субполе «*GeneralizedTime*» может включать разницу между локальным и Гринвичским временем.

В рамках данного стандарта, значения GeneralizedTime-времени должны обязательно отображать время по Гринвичскому (нулевому) меридиану и включать секунды (например, время «*YYMMDDHHMMSSZ*»), и даже в тех слу-

чаях, когда число секунд равно нулю. Значения GeneralizedTime-времени никогда не должны включать доли секунды.

§4.1.2.6 Держатель (владелец) СЕРТ|ОК «Subject»

Поле «*Subject*» идентифицирует субъекта, который связан с открытым ключом, помещённым в поле «*Subject public key*». Имя владельца сертификата может быть расположено в поле «*Subject*» и/или в субполе «*subjectAltName*» (альтернативное имя субъекта) поля «*Расширения*». Если держателем сертификата является УЦ (например, имеет место субполе «*Basic constraints*» в поле «*Расширения*» и значение флага «*cA*» (УЦ) — «*TRUE*», т.е. «подлинный»), то поле «*Subject*» должно обязательно содержать не пустое (*non-empty*) уникальное наименование, с которым сравниваются значения из полей «*Subject*» всех сертификатов, выпущенных этим УЦ, являющимся держателем сертификата. Если держателем сертификата является издатель СОС (например, представлено субполе «*Key usage*» в поле «*Расширения*» и значение в субполе «*cRLSign*» (подписанный СОС) — «*TRUE*», т.е. «подлинный»), то поле «*Subject*» должно обязательно содержать не пустое (*non-empty*) уникальное наименование, с которым сравниваются значения из полей «*Subject*» всех СОС, выпущенных этим издателем, являющимся держателем сертификата. Если информация, именующая владельца сертификата, представлена только в субполе «*subjectAltName*» (альтернативное имя) поля «*Расширения*» (например, ключ привязан только к адресу электронной почты или URI-идентификатору), то имя владельца сертификата должно быть только пустой последовательностью, а субполе «*subjectAltName*» должно иметь статус «критично» (*critical*).

Если поле «*Subject*» не пустое, то оно должно содержать уникальное имя, закодированное в соответствие с правилами Рекомендации X.500 (УИ/X.500). УИ/X.500 должно быть взаимно-однозначным для каждого владельца СЕРТ|ОК, выпущенного одним УЦ, указанным в поле «*Issuer*». УЦ может выпустить несколько сертификатов с одним и тем же УИ/X.500 для одного и того же владельца, указанного в поле «*Subject*».

Поле «*Subject*» содержит наименование («*Name*»), тип которого определён Рекомендацией ITU-T X.501. Требования к внедрению этого поля аналогичны тем, которые определены для поля «*Issuer*». Прикладные системы, основывающиеся на данном стандарте, обязаны принимать имена владельцев сертификатов, включающие те типы атрибутов, которые требуются для поля «*Issuer*» (издатель). Прикладные системы, основывающиеся на данном стандарте, должны быть готовы к приёму имён владельцев сертификатов, включающих рекомендованные для поля «*Issuer*» (издатель) типы атрибутов. Синтаксис и соответствующие OID для таких типов атрибутов представлены в Приложении А. Прикладные системы, основывающиеся на данном стандарте, могут использовать представленные ниже правила сравнения при обработке неизвестных типов атрибутов (т.е. при обработке цепочек имён), значения которых используют одну из дополнительных функций кодирования из правил кодирования данных типа «*DirectoryString*». Двоичное сравнение следует использовать тогда, когда неизвестные типы атрибутов включают значения, закодированные с помощью дополнительных функций, но отличающихся от тех, которые используются для данных типа «*DirectoryString*». Такой подход позволяет прикладным системам обрабатывать сертификаты с неизвестными атрибутами в поле «*Subject*» (имя владельца сертификата).

Когда кодирование атрибутов соответствует правилам кодирования данных типа «*DirectoryString*», УЦ, основывающиеся на данном стандарте, обязаны использовать, либо кодировку «*PrintableString*», либо кодировку «*UTF8String*», со следующими ограничениями:

а) когда владельцем сертификата является УЦ, тогда поле «*Subject*» обязано иметь точно такую же кодировку, которая используется для поля «*Issuer*» во всех сертификатах, изданных этим УЦ (владельцем сертификата). Таким образом, если УЦ, владелец сертификата, кодирует атрибуты в полях «*Issuer*» выпускаемых им сертификатов с помощью *TeletexString*-, *BMPString*- или *UniversalString*-кодировок, то поле «*Subject*» в выпускаемых им сертификатах должно обязательно иметь точно такую же кодировку;

b) когда владельцем сертификата является издатель СОС, тогда поле «*Subject*» обязано иметь точно такую же кодировку, которая используется для поля «*Issuer*» во всех СОС, выпускаемых этим владельцем сертификата;

c) TeletexString-, *BMPString*- и *UniversalString*-кодировки, включённые для обеспечения обратной совместимости, не должны использоваться в сертификатах, выпущенных для новых владельцев. Тем не менее, эти типы могут использоваться в сертификатах в тех случаях, когда имя владельца сертификата было сформировано достаточно давно. К этим случаям относятся также и те, когда издавался новый сертификат для существующего владельца или когда сертификат издавался для нового владельца, но при этом атрибуты кодировались так, как они были сформированы ранее в рамках тех сертификатов, которые были изданы для других владельцев. Пользователи сертификатов должны быть готовы для приёма сертификатов с такими типами кодировок.

Существуют устаревшие прикладные системы, в которых адрес электронной почты выполняет роль уникального имени владельца сертификата в форме атрибута «*emailAddress*» (RFC-2985). Значение атрибута «*emailAddress*» представляет собой тип кодирования «*IA5String*», допускающий использование символа «@», но который не входит в алфавит *PrintableString*-кодировки. Значения атрибута «*emailAddress*» не «чувствительны» к написанию символов в верхнем или нижнем регистре (например, адрес «*subscriber@example.com*» эквивалентен адресу «*SUBSCRIBER@EXAMPLE.COM*»).

Прикладные системы, следующие данному стандарту и формирующие новые сертификаты с адресами электронной почты, с целью установления владельца сертификата обязаны использовать кодирование имени в субполе «*subjectAltName*» (альтернативное имя владельца) в соответствие с правилами стандарта RFC-822. Совместное использование атрибута «*emailAddress*» в составе уникального имени владельца сертификата с целью обеспечения функциональной совместимости устаревших прикладных систем «опротестовано», но тем не менее разрешено.

§4.1.2.7 Информация об открытом ключе владельца СЕРТ|ОК «SubjectPublicKeyInfo»

Это поле предназначено для указания открытого ключа и идентификации криптоалгоритма, совместно с которым используется ключ (например, RSA, DSA или Diffie-Hellman). Криптоалгоритм идентифицируется с помощью субполя «*AlgorithmIdentifier*». OID для определения криптоалгоритмов и способы кодирования данных об открытом ключе (сам открытый ключ и его параметры) представлены в стандартах RFC-3279, RFC-4055 и RFC-4491.

§4.1.2.8 Уникальные идентификаторы «UniqueIdentifier»

Эти дополнительные поля должны присутствовать в сертификате только при использовании 2-ой или 3-ей версии Рекомендации ITU-T X.509. И наоборот, если используется 1-ая версия Рекомендации ITU-T X.509, то указанные поля должны быть исключены из сертификата. Применение УИД владельца и издателя сертификата, представленных в сертификате, позволяет по прошествии некоторого времени повторно использовать имена владельца и/или издателя. Данный стандарт не рекомендует повторное использование наименований (имён) для различных субъектов и использование УИД в Интернет-сертификатах. УЦ, следующие данному стандарту, обязаны не формировать сертификаты с УИД. Прикладные системы, также придерживающиеся данного стандарта, должны проводить синтаксический анализ сертификатов, содержащих УИД, но от них не требуется какая-либо обработка таких УИД.

§4.1.2.9 Расширения «Extensions»

Это поле должно присутствовать в сертификате только при использовании 3-ей версии Рекомендации ITU-T X.509. Если оно представлено, то данное поле представляет собой последовательность из одного или нескольких субполей «*Расширение сертификата*». Формат и содержание данных субполей для Интернет/PKI-инфраструктуры представлены ниже.

§4.2 Субполя «Расширение сертификата» поля «Расширения»

Поле «*Расширения*», описанное в Рекомендации ITU-T X.509 3-ей версии, представляет собой способ, который позволяет «связывать» дополнительные атрибуты с пользователями или открытыми ключами и обеспечивает взаимосвязи между УЦ. Формат сертификата, в соответствие с Рекомендацией ITU-T X.509 3-ей версии, также позволяет включать дополнительные частные субполя «*Расширение сертификата*», которые содержат уникальную корпоративную информацию, необходимую для определённых Интернет-сообществ (частных организаций). Каждое субполе «*Расширение сертификата*» может иметь статус, либо «критичное» (*critical*), либо «не критичное» (*non-critical*). ИТС, использующая сертификаты, обязана удалять сертификат, если она «наталкивается» на «критичное» субполе, но не может его распознать, или «критичное» субполе содержит информацию, которую система не способна обработать. «Не критичное» субполе может быть проигнорировано, если оно не распознаётся, но должно быть обязательно обработано, если оно распознаётся. Далее будут рассмотрены рекомендуемые субполя «*Расширение сертификата*», используемые в Интернет-сертификатах, а также стандартные субполя для размещения соответствующей информации. Частные организации могут использовать и дополнительные субполя, однако, следует проявлять «осторожность» при использовании критичных субполей в сертификатах, так как такие субполя могут препятствовать использованию сертификатов в общем контексте.

Каждое субполе «*Расширение сертификата*» включает OID и последовательность символов в ASN.1-формате. Когда субполе присутствует в сертификате, то OID кодируется как последовательность «*extnID*» (идентификатор расширения), а соответствующая кодовая последовательность октетов «*extnValue*» (содержание расширения) имеет DER/ASN.1-структуру. Сертификат не должен содержать более одного субполя соответствующего расширения. Например, сертификат может содержать только одно субполе расширения «идентификатор ключа УЦ» (*authority key identifier extension*). Субполе «*Расширение сертификата*» включает логическое число «критичности», а в режиме «по умолчанию»

это субполе содержит значение «критичности» — «*FALSE*» (ошибка). Для УЦ, придерживающихся данного стандарта, текст в каждом субполе устанавливает допустимые варианты значения «критичности».

УЦ, придерживающиеся данного стандарта, обязаны использовать субполя следующих расширений сертификата: идентификаторы ключей, основные ограничения, сферы применения ключа и политики сертификации. Если УЦ издаёт сертификаты с пустым полем «*Subject*», то данный УЦ обязан использовать субполе расширения «альтернативное имя владельца сертификата» (*subject alternative name*). Использование остальных субполей в поле «*Расширения*» является необязательным. УЦ, придерживающиеся данного стандарта, также могут применять субполя расширений, которые не представлены в данном стандарте. Издатели сертификатов должны обратить внимание на то, использование не упомянутых в данном стандарте субполей расширений как «критичных» может нарушить функциональную совместимость.

Прикладные системы, базирующиеся на данном стандарте, как минимум обязаны определять следующие субполя расширений: сферы применения ключа, политики сертификации, альтернативное имя владельца сертификата, основные ограничения, ограничения именования, ограничения политики, расширение сферы применения ключа и запрет расширения «*anyPolicy*» (любая политика).

Кроме того, прикладные системы, базирующиеся на данном стандарте, должны определять следующие субполя расширений: УЦ и идентификатор ключа владельца сертификата, а также отображения политики.

§4.2.1 Стандартные субполя поля «Расширения»

Далее рассматриваются стандартные субполя «*Расширение сертификата*», представленные в Рекомендации ITU-T X.509 3-ей версии и предназначенные для использования в Интернет/PKI-инфраструктуре. Каждое такое субполе имеет свой идентификатор, представленный в Рекомендации ITU-T X.509

3-ей версии. Эти OID в ходят в состав архитектуры «*id-ce arc*», определяемой следующим образом:

```
id-ce   OBJECT IDENTIFIER ::=  { joint-iso-ccitt(2) ds(5) 29 } .
```

§4.2.1.1 Идентификатор ключа УЦ «authorityKeyIdentifier»

Это субполе включает параметр для идентификации открытого ключа, соответствующего закрытому ключу, который используется для подписи сертификата. Это субполе используется там, где издатель обладает несколькими ключами для подписи (либо вследствие нескольких равнозначных ключей, либо вследствие перенастройки/переустановки). Такая идентификация может быть основана, либо на идентификаторе ключа (идентификатор ключа владельца, указанный в сертификате издателя), либо на имени издателя и последовательном номере.

Последовательность «*keyIdentifier*» в субполе «*authorityKeyIdentifier*» должна обязательно включаться во все сертификаты, изданные УЦ, придерживающимися данного стандарта, с целью упрощения процедуры формирования МС. Тем не менее, существует одно исключение: если УЦ распространяет свой открытый ключ в форме «само-подписанного» («*self-signed*») сертификата, то субполе «*authorityKeyIdentifier*» может быть пропущено. Подпись на таком сертификате формируется с помощью закрытого ключа, который связан с открытым ключом владельца сертификата. (Это подтверждает, что издатель владеет обоими ключами: открытым и закрытым.) В данном случае, идентификаторы ключей УЦ и владельца сертификата могли бы быть идентичными, но только идентификатор ключа владельца необходим для формирования МС.

Значение последовательности «*keyIdentifier*» должно формироваться из открытого ключа, используемого для проверки подписи сертификата, или способом, используемым для генерирования уникальных чисел. Далее будут рассмотрены два способа формирования идентификаторов ключей из значения открытого ключа. Если идентификатор ключа не был сформирован заранее, то данный стандарт рекомендует использовать один из указанных способов фор-

мирования последовательности «*keyIdentifier*» или использовать аналогичный способ, в котором используется иной алгоритм вычисления хэш-функции. Если же идентификатор ключа был сформирован заранее, то УЦ целесообразно использовать этот идентификатор.

Данный стандарт рекомендует использовать такой подход для всех пользователей сертификатов.

УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «некритичное» («*non-critical*»).

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier             [0] KeyIdentifier             OPTIONAL,
    authorityCertIssuer       [1] GeneralNames              OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber   OPTIONAL  }

KeyIdentifier ::= OCTET STRING
```

§4.2.1.2 Идентификатор ключа владельца сертификата «subjectKeyIdentifier»

Это субполе в поле «Расширения» указывает на средства идентификации сертификатов, которые содержат соответствующий открытый ключ.

В целях упрощения формирования МС данное субполя должно в обязательном порядке присутствовать во всех сертификатах, изданных УЦ, придерживающимися данного стандарта, т.е., во всех сертификатах, включающих субполе «*Basic constraints*» (основные ограничения) поля «Расширения», в которых флаг «*cA*» (УЦ) имеет значение «*TRUE*» («верно»). В сертификатах УЦ, придерживающихся данного стандарта, значение субполя «*subjectKeyIdentifier*» должно в обязательном порядке извлекаться из последовательности «*keyIdentifier*» субполя «*authorityKeyIdentifier*», входящего в состав тех сертификатов, которые были изданы владельцем данного сертификата. Прикладные системы, не требующие проверки таких идентификаторов ключей, сравнивают их при подтверждении подлинности МС.

В сертификатах УЦ значение субполя «*subjectKeyIdentifier*» должно формироваться из открытого ключа, используемого для проверки подписи серти-

фиката, или способом, используемым для генерирования уникальных чисел. Существуют два общих способа формирования идентификаторов ключей из открытого ключа, а именно:

1) последовательность «*keyIdentifier*» формируется из 160-битового результата вычисления хэш-функции стандарта SHA-1 по битовой последовательности (*BIT STRING*) субполя «*subjectKeyIdentifier*» (исключая биты последовательностей «*tag*», «*length*» и другие не используемые биты);

2) последовательность «*keyIdentifier*» формируется из 4-битового поля со значением «*0100*», за которым следует наименее значимые 60 бит результата вычисления хэш-функции стандарта SHA-1 по битовой последовательности (*BIT STRING*) субполя «*subjectPublicKey*» (исключая биты последовательностей «*tag*», «*length*» и другие не используемые биты).

Кроме этого, могут использоваться и другие методы формирования уникальных чисел.

В сертификатах конечных пользователей субполе «*subjectKeyIdentifier*» указывает на средства идентификации сертификатов, которые содержат соответствующий открытый ключ, используемый прикладной системой. Если конечный субъект получил несколько сертификатов, в частности от нескольких УЦ, субполе «*subjectKeyIdentifier*» указывает на средства ускоренной идентификации совокупности сертификатов, содержащих соответствующий открытый ключ. В целях поддержки прикладных систем при идентификации сертификата соответствующего конечного субъекта данной субполе должно включаться во все сертификаты этого конечного субъекта.

В сертификатах конечных субъектов идентификаторы ключей владельцев сертификатов должны формироваться на основе открытого ключа. Ранее были рассмотрены два общих способа формирования идентификаторов ключей из открытого ключа.

Если идентификатор ключа не был сформирован заранее, то данный стандарт рекомендует использовать один из указанных выше способов формирования последовательности «*keyIdentifier*» или использовать аналогичный спо-

соб, в котором используется иной алгоритм вычисления хэш-функции. Если же идентификатор ключа был сформирован заранее, то УЦ целесообразно использовать этот идентификатор.

УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «некритичное» («*non-critical*»).

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 14 }

    SubjectKeyIdentifier ::= KeyIdentifier
```

§4.2.1.3 Предназначение (применимость) ключа «keyUsage»

Субполе «*keyUsage*» в поле «Расширения» устанавливает назначение ключа, представленного в сертификате (например, шифрование, подпись, подпись сертификата). Сферы использования ключа могут быть ограничены, в частности тогда, когда ключ мог бы использоваться в нескольких процедурах, но в конкретной процедуре его использование запрещено. Например, если RSA-ключ должен использоваться только при проверке подписей на электронных документах, не являющихся СЕРТ|ОК или СОС, то биты (флаги) «*digitalSignature*» и/или «*nonRepudiation*» будут установлены в «*1*». Более того, если RSA-ключ должен использоваться только в процедурах обеспечения ключами, то бит (флаг) «*keyEncipherment*» будет установлен в «*1*».

УЦ, придерживающиеся данного стандарта, обязаны включать это субполе в сертификаты, которые содержат открытые ключи, использующиеся при подтверждении подлинности (ПП) ЭЦП на электронных документах, не являющихся СЕРТ|ОК или СОС. УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «критичное» («*critical*»).

```
id-ce-keyUsage     OBJECT IDENTIFIER ::=         { id-ce 15 }

KeyUsage ::=       BIT STRING {
digitalSignature          (0),
nonRepudiation            (1),      -- в последних редакциях стандарта X.509 бит
                                    -- переименован в «contentCommitment»
keyEncipherment           (2),
dataEncipherment          (3),
keyAgreement              (4),
keyCertSign               (5),
```

```
cRLSign          (6),
encipherOnly     (7),
decipherOnly     (8) }
```

Биты, входящие в состав субполя «*keyUsage*» поля «Расширения» используются следующим образом:

1. бит «*digitalSignature*» («*№0*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется для ПП ЭЦП на электронных документах, не являющихся CERT|OK (бит *№5*) или COC (бит *№6*). Данный ключ используется службами аутентификации объекта и источника данных и/или обеспечения целостности;

2. бит «*nonRepudiation*» («*№1*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется для проверки ЭЦП на электронных документах, не являющихся CERT|OK (бит *№5*) или COC (бит *№6*). Данный ключ используется службой неотказуемости, которая обеспечивает защиту от ложного отказа автора подписи от его участия в каком-либо процессе или какой-либо процедуре (действии). Если в дальнейшем возникает конфликтная ситуация, то доверенная третья сторона (ДТС) может определить подлинность и достоверность подписанных данных. (Примечание. В последних редакциях Рекомендации ITU-T X.509 этот бит переименован в «*contentCommitment*».);

3. бит «*keyEncipherment*» («*№2*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется для зашифрования закрытых или секретных ключей, т.е. при доставке ключа. Например, этот бит будет установлен в «*1*», когда открытый RSA-ключ предназначен для зашифрования симметричного ключа, необходимого для расшифрования соответствующих данных, или ассиметричного закрытого ключа;

4. бит «*dataEncipherment*» («*№3*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется для непосредственного зашифрования незащищённых данных пользователя без применения промежуточного симметричного шифрования. (Примечание. Использование этого бита

встречается чрезвычайно редко. Почти все прикладные системы при формировании симметричного ключа используют процедуры доставки или согласования ключа.);

5. бит «*keyAgreement*» («*№4*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется в процедурах согласования ключей. Например, если данный ключ используется в процедуре обеспечения ключами на основе алгоритма Диффи-Хеллмэна (Diffie-Hellman), то этот бит будет установлен в «*1*»;

6. бит «*keyCertSign*» («*№5*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется в процедурах проверки подписей в СЕРТ|ОК. Если бит «*keyCertSign*» установлен в значение «*1*», то и бит «*cA*» субполя «*Basic constraints*» в поле «*Расширения*» должен в обязательном порядке иметь значение «*1*»;

7. бит «*cRLSign*» («*№6*») устанавливается в значение «*1*», когда открытый ключ владельца сертификата используется в процедурах проверки подписей в СОС (например, СОС, усечённый СОС (*delta CRL*) или УЦ-СОС (*ARL*));

8. если отсутствует бит «*keyAgreement*», то значение бита «*encipherOnly*» («*№7*») не установлено. Если же биты «*encipherOnly*» и «*keyAgreement*» одновременно установлены в значение «*1*», то открытый ключ владельца сертификата может использоваться в процессе зашифрования данных, пока проводится процедура согласования ключа;

9. если отсутствует бит «*keyAgreement*», то значение бита «*decipherOnly*» («*№8*») не установлено. Если же биты «*decipherOnly*» и «*keyAgreement*» одновременно установлены в значение «*1*», то открытый ключ владельца сертификата может использоваться в процессе расшифрования данных, пока проводится процедура согласования ключа.

Если в поле «Расширения» присутствует субполе «*keyUsage*», то открытый ключ владельца сертификата не должен, в обязательном порядке, использоваться для проверки подписей в сертификатах и СОС, причём до тех пор, пока соответствующий бит «*keyCertSign*» или «*cRLSign*» не будет установлен в

значение «*1*». Если открытый ключ владельца сертификата предназначен для проверки подписей в сертификатах и/или СОС, то не целесообразно устанавливать биты «*digitalSignature*» и/или «*nonRepudiation*» в значение «*1*». Тем не менее, биты «*digitalSignature*» и/или «*nonRepudiation*» могут быть дополнительно установлены в «*1*», если открытый ключ владельца сертификата предназначен для проверки подписей в сертификатах и/или СОС, а также и в других электронных документах.

Совместное использование бита «*nonRepudiation*» с другими битами субполя «*keyUsage*» в поле «*Расширения*» может означать наличие прикладных аспектов обеспечения безопасности в зависимости от сферы применения сертификата. Последующие различия между битами «*digitalSignature*» и «*nonRepudiation*» могут установлены соответствующими политиками сертификации.

Данный стандарт не устанавливает каких-либо ограничений на комбинации бит, которые могут встречаться при использовании субполя «*keyUsage*» в поле «*Расширения*». Однако, стандарты RFC-3279, RFC-4055 и RFC-4491 в интересах соответствующих криптоалгоритмов устанавливают дополнительные обязательные правила кодирования субполя «*keyUsage*». Когда в поле «*Расширения*» сертификата присутствует субполе «*keyUsage*», тогда, по крайней мере, один бит должен, в обязательном порядке, иметь значение «*1*».

§4.2.1.4 Политики сертификации

Субполе «*certificatePolicies*» в поле «*Расширения*» сертификата содержит последовательность, состоящую из одного или нескольких информативных терминов, описывающих политику сертификации, и каждый из которых состоит из OID и дополнительных определителей. Дополнительные определители, которые могут присутствовать в сертификате, не предполагают внесения изменений в описание политики. OID политики сертификации должен быть представлен в субполе «*certificatePolicies*» поля «*Расширения*» сертификата не более одного раза.

В сертификате конечного пользователя информативные термины, описывающие политику сертификации, указывают на политику, в соответствие с которой выпускается сертификат, и на цели, в соответствие с которыми может использоваться сертификат. В сертификате УЦ информативные термины, описывающие политику сертификации, ограничивают совокупность политик, предназначенных для МС, которые включают данный сертификат. Если УЦ не желает ограничивать совокупность политик, предназначенных для МС, которые включают данный сертификат, то он может указать особую политику «*anyPolicy*» со значением { *2 5 29 32 0* }.

Предполагается, что прикладные системы со специфическими требованиями к политикам сертификации имеют перечень таких приемлемых политик. Они будут сравнивать OID, содержащиеся в сертификатах, с OID, содержащимися в перечнях приемлемых политик сертификации. Если это субполе «*certificatePolicies*» является критичным, то программный модуль проверки подлинности МС обязан корректно интерпретировать это субполе (включая дополнительный определитель), либо уничтожить такой сертификат.

В целях обеспечения функциональной совместимости данный стандарт рекомендует, чтобы информативные термины, описывающие политику сертификации, состояли только из одного OID. В случае, когда одного OID недостаточно, данный стандарт строго рекомендует, чтобы использование определителей было ограничено. Число таких определителей определено ниже. Когда определители используются совместно с особой политикой сертификации «*anyPolicy*», то их число также должно быть ограничено (см. ниже). Принимаются во внимание только те определители, которые возвращаются в качестве результатов выполнения процедуры подтверждения подлинности МС.

Данный стандарт устанавливает два типа определителей политик сертификации, которые используются авторами политик сертификации и издателями сертификатов. К таким определителям относятся определители «*CPS Pointer*» и «*User Notice*» (уведомление пользователя).

Определитель «*CPS Pointer*» содержит указатель на «Официальный отчёт по практической сертификации» (*Certification Practice Statement*), опубликованный УЦ. Указатель представляет собой унифицированный идентификатор Интернет-ресурса (*Uniform Resource Identifier* — URI). Требования к обработке такого определителя являются локальной задачей. В соответствие с данным стандартом данный определитель не должен подвергаться какой-либо обработке, и даже в случае, когда это субполе помечено как «критичное».

Определитель «*User notice*» предназначен для отображения на дисплее проверяющей стороны, когда используется данный сертификат. На дисплее пользователя отображаются только те определители «*User Notice*», которые являются результатами процедуры подтверждения подлинности МС. Если определитель «*User Notice*» был продублирован, то на дисплее следует отображать только одну его копию. Для предотвращения такого дублирования целесообразно, чтобы данный определитель указывался только в сертификатах конечных пользователей и сертификатах УЦ, издаваемых для других организаций.

Определитель «*User Notice*» включает две дополнительные последовательности: «*noticeRef*» и «*explicitText*». Основывающиеся на данном стандарте УЦ не должны использовать последовательность «*noticeRef*».

Последовательность «*noticeRef*», если она используется, определяет наименование организации или идентифицирует, с помощью числового выражения, соответствующий текстовый отчёт, подготовленный этой организацией. Например, рассматриваемая последовательность может идентифицировать организацию «*CertsRUs*» и указывать число «*1*». Как правило, прикладное ПО будет указывать на файл, содержащий действующий набор указаний для «*CertsRUs*». Прикладной процесс извлечёт текстовое указание из файла и разместит его на дисплее. Сообщения могут быть много язычными, что позволит ПО выбрать сообщение на соответствующем языке, приемлемом для данного ПО.

Последовательность «*explicitText*» включает текстовый отчёт, размещаемый непосредственно в сертификате. Эта последовательность представляет со-

бой последовательность символов, состоящей не более чем из 200 символов. Целесообразно, чтобы основывающиеся на данном стандарте УЦ использовали кодировку «*UTF8String*» для данной последовательности, а также обязаны не использовать кодировку «*IA5String*» для данной последовательности. Целесообразно, чтобы последовательность «*explicitText*» не включала каких-либо управляющих символов (например, «*U+0000 to U+001F*» и «*U+007F to U+009F*»). Если в данной последовательности используется кодировка «*UTF8String*» или «*BMPString*», то целесообразно, чтобы все символьные последовательности были нормированы в соответствие с формой нормирования «C» для универсального кода (*Unicode normalization form C* — NFC).

Если в один определитель включены обе последовательности «*noticeRef*» и «*explicitText*», и если ПО способно определить расположение текстового отчёта, указанное последовательностью «*noticeRef*», то этот текстовый отчёт должен быть выведен на дисплей. В противном случае, должна быть выведена на дисплей последовательность «*explicitText*».

(Примечание. Несмотря на то, последовательность «*explicitText*» имеет максимальный размер 200 символов, некоторые УЦ, не придерживающиеся данного стандарта, могут нарушить данное ограничение. По этой причине, пользователям сертификатов рекомендуется обрабатывать последовательность «*explicitText*», длина которой превышает 200 символов.)

```
id-ce-certificatePolicies      OBJECT IDENTIFIER ::=  { id-ce 32 }

anyPolicy                      OBJECT IDENTIFIER ::= { id-ce-certificatePolicies 0 }

certificatePolicies ::=        SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::=          SEQUENCE {
        policyIdentifier               CertPolicyId,
        policyQualifiers               SEQUENCE SIZE (1..MAX) OF
                                       PolicyQualifierInfo OPTIONAL }

CertPolicyId ::=               OBJECT IDENTIFIER

PolicyQualifierInfo ::=        SEQUENCE {
        policyQualifierId              PolicyQualifierId,
        qualifier                      ANY DEFINED BY policyQualifierId }

        -- policyQualifierIds for Internet policy qualifiers
```

```
   id-qt            OBJECT IDENTIFIER ::= { id-pkix 2 }
   id-qt-cps        OBJECT IDENTIFIER ::= { id-qt 1 }
   id-qt-unotice    OBJECT IDENTIFIER ::= { id-qt 2 }

PolicyQualifierId ::=       OBJECT IDENTIFIER ( id-qt-cps | id-qt-unotice )

Qualifier ::=               CHOICE {
        cPSuri                      CPSuri,
        userNotice                  UserNotice }

CPSuri ::=                  IA5String

UserNotice ::=              SEQUENCE {
        noticeRef                   NoticeReference OPTIONAL,
        explicitText                DisplayText OPTIONAL }

NoticeReference ::=         SEQUENCE {
        Organization                DisplayText,
        noticeNumbers               SEQUENCE OF INTEGER }

DisplayText ::=             CHOICE {
        ia5String       IA5String       (SIZE (1..200)),
        visibleString   VisibleString   (SIZE (1..200)),
        bmpString       BMPString       (SIZE (1..200)),
        utf8String      UTF8String      (SIZE (1..200)) }
```

§4.2.1.5 Отображение политик сертификации

Это субполе «*policyMappings*» поля «*Расширения*» используется в сертификатах УЦ. В этом субполе содержатся одна или несколько пар OID. Каждая пара включает последовательности «*issuerDomainPolicy*» и «*subjectDomainPolicy*». Образование пар указывает на то, что выпускающий УЦ рассматривает политику сертификации в своём сегменте «*issuerDomainPolicy*» как эквивалентную политике сертификации, проводимую УЦ в сегменте держателя сертификата «*subjectDomainPolicy*».

Пользователи выпускающих УЦ могут согласиться с политикой «*issuerDomainPolicy*» для определённых прикладных систем. Субполе «*policyMappings*» определяет перечень политик, связанных с УЦ владельца сертификата, которые могут рассматриваться как сопоставимые с политикой «*issuerDomainPolicy*».

Каждая указанная в субполе «*policyMappings*» политика «*issuerDomainPolicy*» также должна указываться в субполе «*certificatePolicies*» в рамках одного и того же сертификата. Политики сертификации не должны в обязательном

порядке отображаться в особую политику сертификации «*anyPolicy*» или наоборот.

В целом, политики сертификации, которые представлены в последовательности «*issuerDomainPolicy*» субполя «*policyMappings*», не рассматриваются как политики для включения в последующие сертификаты на МС. В некоторых случаях, УЦ могут пожелать отобразить одну политику (*p1*) в другую (*p2*), но по-прежнему будут хотеть, чтобы политика «*issuerDomainPolicy*» (*p1*) рассматривалась как приемлемая для включения в последующие сертификаты. Такое событие может произойти, например, если УЦ находится в процессе переходного периода, т.е. переходит от использования политики *p1* к использованию политики *p2*, и при этом имеет достоверные сертификаты, которые были выпущены в соответствие с каждой из политик. УЦ может указать на это путём включения двух субполей «*policyMappings*» в сертификаты УЦ, которые он издал. Каждое субполе «*policyMappings*» могло бы включать политику «*issuerDomainPolicy*» из *p1*, одно субполе «*policyMappings*» могло бы включать политику «*subjectDomainPolicy*» из *p1*, а другое могло бы включать политику «*subjectDomainPolicy*» из *p2*.

Это субполе может применяться УЦ и/или прикладными системами. УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «критичное» («*critical*»).

```
id-ce-policyMappings        OBJECT IDENTIFIER ::=  { id-ce 33 }

PolicyMappings ::=          SEQUENCE SIZE (1..MAX) OF SEQUENCE {
            issuerDomainPolicy              CertPolicyId,
            subjectDomainPolicy             CertPolicyId }
```

§4.2.1.6 Альтернативное имя владельца сертификата

Субполе «*subjectAltName*» поля «Расширения» позволяет «привязать» параметры подлинности к держателю сертификата. Эти параметры подлинности могут быть включены дополнительно или вместо параметра подлинности, расположенного в поле «*Subject*» сертификата. К некоторым дополнительным параметрам относятся адрес электронной Интернет-почты, DNS-имя, IP-адрес и

URI-идентификатор. Существуют и другие дополнительные параметры, к которым относятся только локальные определения. Сюда могут относиться несколько форм имён и несколько вариантов каждой из форм имён. Всякий раз когда такие идентификаторы должны быть «привязаны» к сертификату, субполе с альтернативным именем владельца сертификата (или альтернативным именем издателя сертификата) должны использоваться в обязательном порядке. Тем не менее, в поле «*Subject*», используя атрибут «*domainComponent*», может быть также представлено и DNS-имя. (Примечание. Если в поле «*Subject*» представлены такие имена, то дополнительные программные модули для преобразования таких имён в DNS-имена не требуются.)

Так как считается, по определению, что альтернативное имя владельца сертификата должно «привязываться» к открытому ключу, все части альтернативного имени владельца сертификата должны в обязательном порядке проверяться УЦ.

Более того, если только в сертификате представлен параметр подлинности владельца сертификата в форме альтернативного имени (например, адрес электронной почты), то уникальное имя владельца сертификата в обязательном порядке должно быть не пустым (не пустая последовательность), а в сертификате в обязательном порядке должно быть представлено субполе «*subjectAltName*». Если поле «*Subject*» содержит пустую последовательность, то выпускающий УЦ обязан включить в сертификат субполе «*subjectAltName*» и пометить его как «критичное». Если в сертификат включено субполе «*subjectAltName*», а сам сертификат содержит не пустое уникальное имя держателя сертификата, то УЦ, придерживающиеся данного стандарта, должны помечать данное субполе как «не критичное».

Когда субполе «*subjectAltName*» содержит адрес Интернет-почты, то данный адрес должен иметь формат, представленный в стандарте RFC-822 («*rfc822Name*»). Формат «*rfc822Name*» представляет собой формат адреса почтового ящика («*Mailbox*»), определённый в стандарте RFC-2821. Формат адреса почтового ящика имеет вид «*Local-part@Domain*». (Примечание. Адрес почто-

вого ящика не включает перед собой какой-либо фразы (например, стандартного имени) и не имеет после себя комментария (текса в скобках), а также не обрамляется символами «<» и «>».) Правила кодирования адресов Интернет-почты, которые включают международные имена сетевых сегментов/областей, рассматриваются ниже.

Когда субполе «*subjectAltName*» содержит IP-адрес, тогда этот адрес должен иметь формат последовательности октетов, расположенных в сетевом порядке для передачи. Указанный формат представлен в стандарте RFC-791. Наименее значимый бит (*the least significant bit* — LSB) представляет собой LSB-бит соответствующего байта сетевого адреса. Для IP-адреса четвёртой версии (IPv4-адрес, RFC-791) последовательность октетов должна содержать строго четыре байта. Для IP-адреса шестой версии (IPv6-адрес, RFC-2460) последовательность октетов должна содержать строго шестнадцать байт.

Когда субполе «*subjectAltName*» содержит маркер DNS-имени, тогда альтернативное имя владельца сертификата в обязательном порядке должно иметь кодировку «*IA5String*» в соответствующем *dNSName*-формате. DNS-имя должно иметь предпочтительный синтаксис («*preferred name syntax*», RFC-1034), или модифицированный синтаксис в соответствие со стандартом RFC-1123. Следует отметить, что в DNS-именах допускаются символы в обоих регистрах (строчные и ПРОПИСНЫЕ), но в данном случае это не имеет значения. Более того, если для DNS-имён разрешается использовать пустую последовательность «...», то в субполе «*subjectAltName*» такие последовательности использовать запрещено. И в заключении, использование DNS-имён в качестве адресов Интернет-почты («*subscriber.example.com*» вместо «*subscriber@example.com*») категорически запрещено. Такие параметры подлинности должны кодироваться в *rfc822Name*-формате. Правила кодирования международных имён сетевых сегментов/областей, рассматриваются ниже.

Когда субполе «*subjectAltName*» содержит URI-идентификатор, тогда альтернативное имя владельца сертификата в обязательном порядке должно иметь кодировку «*IA5String*» в соответствующем *uniformResourceIdentifier*-формате.

Для альтернативного имени запрещено использовать относительный URI-идентификатор. Оно должно формироваться в соответствие URI-синтаксисом и правилами кодирования, представленными в стандарте RFC-3986. Альтернативное имя должно включать обе схемы идентификации: схема адресации (например, «*http*» или «*ftp*») и схема со специфическим кодированием частей адреса («*scheme-specific-part*»). URI-идентификаторы, которые включают наименование УЦ (RFC-3986), должны включать в обязательном порядке полностью определённое имя сегмента/области или IP-адрес, как у IP-узла. Правила кодирования международных идентификаторов Интернет-ресурсов рассматриваются ниже.

Как определено в стандарте RFC-3986, название схемы адресации не зависит от регистра написания символов (например, «*http*» эквивалент «*HTTP*»). Название схемы со специфическим кодированием частей адреса, если оно представлено, также не зависит от регистра написания символов, но другие компоненты адреса со специфическим кодированием его частей могут зависеть от регистра написания символов. Правила сравнения URI-идентификаторов рассматриваются ниже.

Когда субполе «*subjectAltName*» поля «*Расширения*» содержит имя Службы единого каталога (СЕК, *The Directory*) в формате «*directoryName*», тогда правила кодирования альтернативного имени владельца сертификата аналогичны правилам, которые установлены для поля «*Issuer*» (§4.1.2.4). СЕК-имя должно быть уникальным для каждого держателя сертификата и заверенным тем же УЦ, как это определено в поле «*Issuer*». УЦ может издать несколько сертификатов с одним и тем же СЕК-именем и для одного и того же владельца сертификата.

Субполе «*subjectAltName*» может содержать дополнительные типы наименований путём использования последовательности «*otherName*». Формат и семантика такого наименования указывается с помощью OID, расположенного в субпоследовательности «*type-id*». Само же имя располагается в субпоследовательности «*value*» последовательности «*otherName*». Например, имена в

Kerberos-формате стандарта RFC-4120 могут быть закодированы в последовательности «*otherName*» с использованием OID («*type-id*»), определяющего имя участника информационного обмена согласно пятой версии *Kerberos*-стандарта, и последовательности символов («*value*»), состоящей из наименования зоны («*Realm*») и имени участника информационного обмена («*PrincipalName*»).

Конструкции альтернативных имён владельцев сертификатов могут иметь ограничения по аналогии с уникальными именами владельцев сертификатов на основе субполя «*nameConstraints*» (§4.2.1.10).

Если субполе «*subjectAltName*» представлено, то в нём должна обязательно присутствовать хотя бы одна последовательность (строка). В отличие от поля «*Subject*», придерживающиеся данного стандарта УЦ не должны выпускать сертификаты с субполями «*subjectAltName*», включающими пустые последовательности «*GeneralName*». Например, имя в *rfc822Name*-формате должно иметь кодировку «*IA5String*». Несмотря на то, что пустая последовательность может иметь допустимую кодировку «*IA5String*», использовать имя в *rfc822Name*-формате данным стандартом запрещено. Поведение клиентов, которые «наталкиваются» на такие сертификаты при обработке МС, данным стандартом не определено.

И в заключении, семантика альтернативных имён владельцев сертификатов, которые включают произвольные (*wildcard*) символы (например, «заполнитель» (структурный ноль) для совокупности имён), в данном стандарте не рассматривается. Прикладным системам со специфическими требованиями разрешено использовать указанные имена, но в таких случаях они обязаны указывать семантику.

```
id-ce-subjectAltName    OBJECT IDENTIFIER ::=    { id-ce 17 }

SubjectAltName ::=      GeneralNames

GeneralNames ::=        SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::=         CHOICE {
        otherName                   [0]     OtherName,
        rfc822Name                  [1]     IA5String,
        dNSName                     [2]     IA5String,
        x400Address                 [3]     ORAddress,
```

```
        directoryName                [4]    Name,
        ediPartyName                 [5]    EDIPartyName,
        uniformResourceIdentifier    [6]    IA5String,
        iPAddress                    [7]    OCTET STRING,
        registeredID                 [8]    OBJECT IDENTIFIER }

OtherName ::=           SEQUENCE {
        type-id                      OBJECT IDENTIFIER,
        value                        [0]    EXPLICIT ANY DEFINED BY type-id }

EDIPartyName ::=        SEQUENCE {
        nameAssigner                 [0]    DirectoryString OPTIONAL,
        partyName                    [1]    DirectoryString }
```

§4.2.1.7 Альтернативное имя издателя сертификата

Субполе «*issuerAltName*» поля «*Расширения*», как и субполе «*subjectAltName*», используется для «привязки» параметров подлинности, применяемых в Интернет-сети, к издателю сертификата. Субполе «*issuerAltName*» должно кодироваться по правилам, представленным в §4.2.1.6. Альтернативные имена издателей не должны обрабатываться в рамках процедуры (как её часть) подтверждения подлинности МС. (Т.е., альтернативные имена издателей не используются в цепочке имён МС, а также на них распространяются ограничения, налагаемые на конструкции имён.)

Если в сертификат включено субполе «*issuerAltName*», то УЦ, придерживающиеся данного стандарта, должны помечать это субполе как «не критичное».

```
id-ce-issuerAltName     OBJECT IDENTIFIER ::= { id-ce 18 }

IssuerAltName ::=       GeneralNames
```

§4.2.1.8 Атрибуты СЕК-сегмента владельца сертификата

Субполе «*subjectDirectoryAttributes*» поля «*Расширения*» используется для доставки идентификационных атрибутов владельца сертификата (например, гражданство). Это субполе представляет собой последовательность из одного или нескольких атрибутов. УЦ, придерживающиеся данного стандарта, должны помечать это субполе как «не критичное».

```
id-ce-subjectDirectoryAttributes        OBJECT IDENTIFIER ::= { id-ce 9 }
```

SubjectDirectoryAttributes ::= SEQUENCE SIZE (1..MAX) OF Attribute

§4.2.1.9 Основные ограничения

Субполе «*basicConstraints*» поля «*Расширения*» определяет является ли УЦ владельцем данного сертификата, а также максимальную «протяжённость» (глубину) приемлемого МС, который включает данный сертификат.

Установленный в «*1*» бит (флаг) «*cA*» указывает может ли сертифицированный открытый ключ использоваться для проверки подписей на сертификатах. Если флаг «*cA*» не установлен, то и бит «*keyCertSign*» в субполе «*keyUsage*» тоже должен быть нулевым. Если субполе «*basicConstraints*» не представлено в третьей версии сертификата, или оно представлено, но не установлен флаг «*cA*», то сертифицированный открытый ключ не должен использоваться для проверки подписей на сертификатах.

Последовательность «*pathLenConstraint*» (ограничение протяжённости МС) имеет смысл только тогда, когда установлен флаг «*cA*» и субполе «*keyUsage*», если оно представлено, содержит установленный в «*1*» бит «*keyCertSign*». В данном случае, эта последовательность указывает на максимальное число не само-изданных промежуточных сертификатов, которые могут следовать за этим сертификатом по приемлемому МС. (Примечание. Последний сертификат на МС не является промежуточным сертификатом, и не является его пределом. Обычно, последний сертификат является сертификатом конечного пользователя, но может быть сертификатом УЦ.) Последовательность «*pathLenConstraint*» с нулевым значение указывает на то, что на МС не могут следовать не само-изданные промежуточные сертификаты УЦ. Если такой случай имеет место, то последовательность «*pathLenConstraint*» должна в обязательном порядке содержать значение равное или больше нуля. Если последовательность «*pathLenConstraint*» отсутствует, то ограничение не устанавливается.

УЦ, придерживающиеся данного стандарта, обязаны включать субполе «*basicConstraints*» во все сертификаты УЦ, содержащие открытые ключи, используемые для подтверждения подлинности цифровых подписей в сертифика-

тах, и кроме того, в указанных сертификатах такие УЦ обязаны помечать это субполе как «критичное». Субполе «*basicConstraints*» может присутствовать в сертификатах УЦ как «критичное» или как «некритичное», если такие сертификаты содержат открытые ключи, используемые исключительно в тех целях, которые отличны от подтверждения подлинности цифровых подписей в сертификатах. К сертификатам таких УЦ относятся, либо сертификаты, которые содержат открытые ключи, используемые исключительно в процедурах подтверждения подлинности цифровых подписей в СОС, либо сертификаты, которые содержат открытые ключи, используемые в процедурах обеспечения криптоключами в соответствие с протоколами, зарегистрированными в сертификатах. Субполе «*basicConstraints*» может присутствовать в сертификатах конечных пользователей как «критичное» или как «некритичное».

УЦ запрещено включать последовательность «*pathLenConstraint*» до тех пор, пока не будет установлен флаг «*cA*», а в субполе «*keyUsage*» не будет установлен флаг «*keyCertSign*».

```
id-ce-basicConstraints      OBJECT IDENTIFIER ::= { id-ce 19 }

BasicConstraints ::=        SEQUENCE {
        cA                              BOOLEAN DEFAULT FALSE,
        pathLenConstraint               INTEGER (0..MAX) OPTIONAL }
```

§4.2.1.10 Ограничения на форматы имён

Субполе «*nameConstraints*», которое может использоваться только в сертификатах УЦ, указывает на пространство имён, в рамках которого все имена владельцев сертификатов в последовательности сертификатов на пути сертификации должны быть в обязательном порядке обнаруживаемыми (устанавливаемыми). Ограничения накладываются на уникальное и альтернативные имена владельца сертификата. Ограничения накладываются только тогда, когда имеет место определённый формат наименования. Если в сертификате нет имени разрешённого типа (формата), то сертификат считается не приемлемым (не допустимым).

Ограничения на форматы имён (субполе «*nameConstraints*» поля «*Расширения*») не применяются в само-изданных сертификатах (до тех пор, пока сертификат не является финальным сертификатом на МС). (Такой подход мог бы предотвратить применение УЦ, использующими ограничения на форматы имён, само-изданных сертификатов для многократного продления срока действия криптоключа.)

Ограничения определяются в терминах субдеревьев имён (ветвей дерева имён), которые разрешены или запрещены к использованию. Любое имя, совпадающее с ограничением, которое указанном в последовательности «*excludedSubtrees*», является недопустимым, даже невзирая на информацию, содержащуюся в последовательности «*permittedSubtrees*». УЦ, придерживающиеся данного стандарта, обязаны помечать субполе «*nameConstraints*» как «критичное», а также не должны налагать ограничений на имена в форматах «*x400Address*», «*ediPartyName*» или «*registeredID*». УЦ, придерживающиеся данного стандарта, обязаны не издавать сертификаты, в которых субполе «*nameConstraints*» представлено в виде пустой (нулевой) последовательности. Т.е., в сертификатах обязательно должна быть, как минимум, одна из двух последовательностей, либо «*excludedSubtrees*», либо «*permittedSubtrees*».

Прикладные системы, придерживающиеся данного стандарта, обязаны обрабатывать субполе «*nameConstraints*», которое налагает ограничения на имена в формате «*directoryName*». И целесообразно, чтобы они обрабатывали субполе «*nameConstraints*», которое налагает ограничения на имена в форматах «*rfc822Name*», «*uniformResourceIdentifier*», «*dNSName*» и «*iPAddress*». Если субполе «*nameConstraints*», помеченное как «критичное», налагает ограничения на соответствующий формат имён, а элемент имени в таком формате представлен в поле «*Subject*» или в субполе «*subjectAltName*» данного сертификата, то прикладная система обязана, либо обработать субполе «*nameConstraints*», либо уничтожить сертификат.

В данном стандарте последовательности «*minimum*» и «*maximum*» не используются ни при каких форматах имён. Таким образом, минимальное значе-

ние должно быть нулевым, а максимальное — вообще отсутствовать. Однако, если прикладная система с «критичным» субполем «*nameConstraints*», в котором указаны иные значения минимума или максимума для форматов имён, представленный в данном сертификате, то прикладная система обязана, либо обработать эти субполе и последовательности, либо уничтожить сертификат.

При использовании URI-идентификаторов ограничение накладывается на часть имени (адреса) IP-узла. Такое ограничение должно в обязательном порядке указывать на полностью определённое наименование сегмента/области и может указывать на IP-узел или сетевой сегмент (область). Примерами сказанного выше могли бы быть адреса «*host.example.com*» и «*.example.com*». Если ограничение начинается с точки, разделяющей маркеры, то оно может быть продолжено на один или несколько маркеров. Т.е., под ограничение «*.example.com*» подпадают два следующих адреса: «*host.example.com*» и «*my.host.example.com*». Тем не менее, под ограничение «*.example.com*» не подпадает адрес «*example.com*». Если ограничение не начинается с точки, разделяющей маркеры, то оно указывает на адрес IP-узла. Если ограничение накладывается на формат имён «*uniformResourceIdentifier*», а в данном сертификате присутствует субполе «*subjectAltName*», содержащее последовательность «*uniformResourceIdentifier*», которая, в свою очередь, не содержит компонент «*Authority*» с наименованием IP-узла в формате полностью определённого наименования сегмента/области (например, если URI-идентификатор, либо не содержит компонент «*Authority*», либо содержит компонент «*Authority*», в котором наименование IP-узла представлено в формате IP-адреса), то прикладная система обязана уничтожить такой сертификат.

Ограничение на имена в формате адресов электронной Интернет-почты могут указывать на соответствующий почтовый ящик, на все адреса соответствующего IP-узла (почтового сервера) или на все почтовые ящики в сетевом сегменте (области). Для указания соответствующего почтового ящика ограничение (субполе «*nameConstraints*» поля «*Расширения*») содержит полный адрес электронной почты. Например, адрес «*root@example.com*» определяет корневой

почтовый ящик, расположенный на IP-узле (почтовом сервере) с именем «*example.com*». Для указания всех адресов электронной Интернет-почты на соответствующем IP-узле (почтовом сервере) ограничение (субполе «*nameConstraints*» поля «*Расширения*») содержит наименование этого IP-узла (почтового сервера). Например, ограничение «*example.com*» накладывает запрет на использование любого адреса почтовой службы относящегося к IP-узлу (почтовому серверу) с именем «*example.com*». Для указания любого адреса почтовой службы в границах сетевого сегмента (области) ограничение начинается с точки, разделяющей маркеры (по аналогии с URI-идентификаторами). Например, под ограничение «.*example.com*» попадают все адреса Интернет-почты в границах сетевого сегмента (области) «*example.com*», но не адреса Интернет-почты, относящиеся к IP-узлу (почтовому серверу) с именем «*example.com*».

Ограничения на DNS-имена отображаются как «*host.example.com*». Любое DNS-имя, которое может быть сформировано путём обычного добавления маркеров с левой стороны имени, подпадает под данное ограничение. Например, DNS-имя «*www.host.example.com*» могло бы попасть под данное ограничение, а вот DNS-имя «*host1.example.com*» не могло бы.

Существуют устаревшие прикладные системы, в которых адрес электронной почты встроен в уникальное имя владельца сертификата в качестве атрибута типа «*emailAddress*» (§4.1.2.6). Когда на имена формата «*rfc822Name*» наложено ограничение, но сертификат не включает альтернативное имя владельца сертификата (субполе «*subjectAltName*» поля «*Расширения*»), тогда на атрибут типа «*emailAddress*» в составе уникального наименования владельца сертификата должно в обязательном порядке распространяться ограничение на имена формата «*rfc822Name*». В Приложении А представлен ASN.1-синтаксис для атрибута типа «*emailAddress*» и соответствующего OID.

Ограничения на имена формата «*directoryName*» обязаны налагаться на поле «*Subject*» в сертификате (если сертификат содержит не пустое поле «*Subject*») и на любые наименования типа «*directoryName*» в субполе «*subjectAlt-*

Name». Ограничения на имена формата «*x400Address*» обязаны налагаться на любые наименования типа «*x400Address*» в субполе «*subjectAltName*».

Если на имена формата «*directoryName*» наложены ограничения, то прикладной программный модуль обязан сравнивать DN-атрибуты. Как минимум, прикладные системы (программные модули) обязаны «выполнять» правила сравнения СЕК-имён, которые рассматриваются ниже. УЦ, выпускающим сертификаты с ограничениями на имена формата «*directoryName*», не целесообразно полагаться на реализацию алгоритма сравнения полных ISO/DN-имён. Это означает, что ограничения на форматы имён должны в обязательном порядке точно совпадать с кодированием, которое используется в поле «*Subject*» или в субполе «*subjectAltName*».

Синтаксис IP-адресов должен быть точно таким же, как он представлен в §4.2.1.6. И вместе с тем, специально для ограничений форматов имён существуют некоторые дополнения. При использовании IPv4-адресов субпоследовательность «*iPAddress*» в последовательности «*GeneralName*» должна в обязательном порядке включать восемь (8) октетов, которые закодированы в формате, определённом в стандарте RFC-4632, предназначенном для описания диапазонов IP-адресов. При использовании IPv6-адресов субпоследовательность «*iPAddress*» в последовательности «*GeneralName*» должна в обязательном порядке включать 32 октета, которые закодированы аналогичным способом. Например, ограничение на формат имён для подсети «*192.0.2.0*» класса *C* будет иметь вид следующей последовательности октетов «*C0 00 02 00 FF FF FF 00*», т.е. маска подсети «*255.255.255.0*», а префиксная форма — «*192.0.2.0/24*».

Дополнительные правила кодирования и обработки субполе «*nameConstraints*» будут представлены ниже.

Синтаксис и семантика ограничений для форматов имён «*otherName*» «*ediPartyName*» и «*registeredID*» в данном стандарте не рассматриваются. Тем не менее, синтаксис и семантика ограничений для иных форматов имён могут быть представлены в других документах.

```
id-ce-nameConstraints      OBJECT IDENTIFIER ::= { id-ce 30 }

NameConstraints ::=        SEQUENCE {
        permittedSubtrees      [0]     GeneralSubtrees OPTIONAL,
        excludedSubtrees       [1]     GeneralSubtrees OPTIONAL }

GeneralSubtrees ::=        SEQUENCE SIZE (1..MAX) OF GeneralSubtree
GeneralSubtree ::=         SEQUENCE {
        base                   GeneralName,
        minimum                [0]     BaseDistance DEFAULT 0,
        maximum                [1]     BaseDistance OPTIONAL }

BaseDistance ::=           INTEGER (0..MAX)
```

§4.2.1.11 Ограничения на применение политик

Субполе «*policyConstraints*» поля «*Расширения*» может использоваться в сертификатах, выпускаемых для УЦ. Это субполе ограничивает подтверждение подлинности МС двумя способами: либо оно запрещает отображение политик, либо оно содержит требование, чтобы каждый сертификат в МС включал приемлемый идентификатор политики.

Если имеет место последовательность «*inhibitPolicyMapping*», то её значение указывает на число дополнительных сертификатов, которые могут присутствовать на МС ещё до того, как отображение политики не будет больше разрешено. Например, значение «*1*» указывает на то, что отображение политики может обрабатываться, но только в тех сертификатах, которые были изданы держателем данного сертификата, а не в дополнительных сертификатах, присутствующих на маршруте.

Если имеет место последовательность «*requireExplicitPolicy*», то её значение указывает на число дополнительных сертификатов, которые могут присутствовать на МС прежде, чем потребуется точно определённая политика для всего МС. Если требуется точно определённая политика, то необходимо, чтобы она была включена во все сертификаты, присутствующие на маршруте, в формате идентификатора приемлемой политики в составе субполя «*certificatePolicies*» поля «*Расширения*» сертификата. Идентификатор приемлемой политики представляет собой идентификатор политики, которая востребована пользователем МС, или идентификатор политики, которая была объявлена эквивалентной на основе отображения политики.

УЦ, придерживающиеся данного стандарта, обязаны обрабатывать последовательность «*requireExplicitPolicy*», а также им целесообразно обрабатывать последовательность «*inhibitPolicyMapping*». Прикладные системы, которые обрабатывают последовательность «*inhibitPolicyMapping*», также обязаны обрабатывать субполе «*policyConstraints*» в поле «*Расширения*» сертификата. Если субполе «*policyConstraints*» помечено, как «критичное», и в нём представлена последовательность «*inhibitPolicyMapping*», то прикладные системы, которые не способны обрабатывать последовательность «*inhibitPolicyMapping*», обязаны удалять такие сертификаты.

УЦ, придерживающиеся данного стандарта, обязаны не издавать сертификаты, в которых субполе «*policyConstraints*» является пустой последовательностью. Т.е., в субполе «*policyConstraints*» должна быть представлена, либо последовательность «*inhibitPolicyMapping*», либо последовательность «*requireExplicitPolicy*». Поведение клиентов, которые натолкнулись на пустое субполе «*policyConstraints*», в данном стандарте не рассматривается.

УЦ, придерживающиеся данного стандарта, должны помечать это субполе как «критичное».

```
id-ce-policyConstraints      OBJECT IDENTIFIER ::=  { id-ce 36 }

PolicyConstraints ::=        SEQUENCE {
        requireExplicitPolicy            [0]     SkipCerts OPTIONAL,
        inhibitPolicyMapping             [1]     SkipCerts OPTIONAL }

SkipCerts ::=                INTEGER (0..MAX)
```

§4.2.1.12 Расширение сферы применения ключа

Это субполе «*extKeyUsage*» в поле «*Расширения*» указывает на одну или несколько сфер, в которых может использоваться сертифицированный открытый ключ, причём в дополнение или вместо основных сфер применения, представленных в субполе «*keyUsage*» поля «*Расширения*». Как правило, это субполе «*extKeyUsage*» будет присутствовать только в сертификатах конечных пользователей. Данное субполе имеет следующий формат:

```
id-ce-extKeyUsage          OBJECT IDENTIFIER ::= { id-ce 37 }
ExtKeyUsageSyntax ::=      SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId ::=           OBJECT IDENTIFIER
```

При необходимости любая организация может определять сферы применения ключа. OID, используемые для идентификации сфер применения ключа, должны в обязательном порядке назначаться в соответствие с правилами IANA или Рекомендацией ITU-T X.660.

Это субполе «*extKeyUsage*», в зависимости от решения издателя сертификата, может помечаться, либо как «критичное», либо как «не критичное».

Если это субполе представлено, то сертификат должен в обязательном порядке использоваться только в одной из указанных сфер применения. Если указано несколько сфер применения, то прикладной системе нет необходимости распознавать все указанные сферы применения, поскольку целевая сфера применения представлена. Прикладные системы, пользующиеся сертификатами, могут потребовать, чтобы в сертификатах было представлено субполе «*extKeyUsage*», в котором была указана соответствующая сфера применения ключа. Такое требование вызвано обеспечением доступности сертификатов, которыми пользуются прикладные системы.

Если УЦ включают в сертификаты субполе «*extKeyUsage*» для удовлетворения требований таких прикладных систем, но в то же время не желают ограничивать сферы применения ключа, то УЦ могут включать специальную субпоследовательность «*KeyPurposeId*» в последовательности «*anyExtendedKeyUsage*» в качестве дополнения к соответствующим сферам применения ключа, которые востребованы прикладными системами. Целесообразно, чтобы УЦ, придерживающиеся данного стандарта, не помечали данное субполе как «критичное», при условии наличия в нём специальной субпоследовательности «*KeyPurposeId*» в последовательности «*anyExtendedKeyUsage*». Прикладные системы, требующие наличия в сертификате соответствующей сферы применения

ключа, могут удалять сертификаты, которые содержат объектный идентификатор «*anyExtendedKeyUsage*», но не OID, ожидаемый прикладной системой.

Если сертификат содержит два субполя «*keyUsage*» и «*extKeyUsage*» одновременно, то оба субполя должны в обязательном порядке обрабатываться независимо друг от друга, а сертификат должен в обязательном порядке использоваться только в той сфере применения, которая указана в обоих субполях. Если в обоих субполях не указана сфера применения, то сертификат не должен никогда использоваться в какой-либо иной сфере применения. Далее представлен формат кодирования последовательности «*anyExtendedKeyUsage*»:

```
anyExtendedKeyUsage     OBJECT IDENTIFIER ::=       { id-ce-extKeyUsage 0 }

id-kp                   OBJECT IDENTIFIER ::=       { id-pkix 3 }

id-kp-serverAuth        OBJECT IDENTIFIER ::=       { id-kp 1 }
                            -- Аутентификация W3-сервера на основе TLS-протокола
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature», «keyEncipherment» или
                            -- «keyAgreement»

id-kp-clientAuth        OBJECT IDENTIFIER ::=       { id-kp 2 }
                            -- Аутентификация W3-клиента на основе TLS-протокола
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature», и/или «keyAgreement»

id-kp-codeSigning       OBJECT IDENTIFIER ::=       { id-kp 3 }
                            -- Подпись загружаемого исполняемого кода
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature»

id-kp-emailProtection   OBJECT IDENTIFIER ::=       { id-kp 4 }
                            -- Защита электронной почтовой службы
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature», «nonRepudiation»
                            -- и/или («keyEncipherment» или «keyAgreement»)

id-kp-timeStamping      OBJECT IDENTIFIER ::=       { id-kp 8 }
                            -- Привязка результата вычисления хэш-функции
                            -- по последовательности данных к времени
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature» и/или «nonRepudiation»

id-kp-OCSPSigning       OBJECT IDENTIFIER ::=       { id-kp 9 }
                            -- Подпись OCSP-ответов (Online Certificate Status
                            -- Protocol, интерактивный протокол определения
                            -- состояния сертификата)
                            -- Биты субполя «keyUsage», которые могут быть
                            -- установлены: «digitalSignature» и/или «nonRepudiation»
```

§4.2.1.13 Точки распространения СОС

Субполе «*cRLDistributionPoints*» указывает на то, как можно получить информацию о СОС. Целесообразно, чтобы это субполе помечалось как «не критичное», но данный стандарт рекомендует обрабатывать данное субполе всеми УЦ и прикладными системами. Проблемы обеспечения СОС рассматриваются ниже.

Субполе «*cRLDistributionPoints*» представляет собой набор следующих друг за другом последовательностей «*DistributionPoint*». Последовательность «*DistributionPoint*» состоит из трёх субпоследовательностей, каждая из которых является дополнительной (не обязательной): «*distributionPoint*», «*reasons*» и «*cRLIssuer*». Несмотря на то, что субпоследовательности являются не обязательными, последовательность «*DistributionPoint*» не должна содержать только одну субпоследовательность «*reasons*», но в ней должна быть в обязательном порядке представлена, либо субпоследовательность «*distributionPoint*», либо субпоследовательность «*cRLIssuer*». Если издатель сертификата не является издателем СОС, то обязательно должна быть представлена субпоследовательность «*cRLIssuer*», содержащая имя издателя СОС. Если издатель сертификата также является издателем СОС, то УЦ, придерживающиеся данного стандарта, обязаны пропускать субпоследовательность «*cRLIssuer*», но обязаны включать субпоследовательность «*distributionPoint*».

Если имеет место субпоследовательность «*distributionPoint*», то она должна содержать, либо набор следующих друг за другом общих имён, либо одиночное значение «*nameRelativeToCRLIssuer*». Если последовательность «*DistributionPoint*» включает несколько величин, то каждое имя определяет специфический способ получения одного и того же СОС. Например, один и тот же СОС мог быть доступен при поиске с помощью LDAP- и HTTP-протоколов.

Если субпоследовательность «*distributionPoint*» содержит имя в формате «*directoryName*» (СЕК-имя), то запись под таким именем «*directoryName*» включает действующий СОС по указанным в субпоследовательности «*reasons*» причинам и СОС, выпущенный указанным в субпоследовательности «*cRLIssuer*» издателем. СОС может содержаться, либо в атрибуте «*certificateRevoca-*

tionList», либо в атрибуте «*authorityRevocationList*». Прикладная система (прикладной процесс) может получить СОС из любого СЕК-сервера, настраиваемого локально. Протокол, который использует прикладная система (прикладной процесс) для обращения к Службе единого каталога (например, DAP- или LDAP-протокол), выбирается локально самой прикладной системой.

Если последовательность «*DistributionPointName*» в субполе «*cRLDistributionPoints*» включает общее имя в URI-формате, то следует в обязательном порядке применять следующую семантику: URI-идентификатор является указателем на действующий СОС по указанным в субпоследовательности «*reasons*» причинам и СОС будет выпускаться указанным в субпоследовательности «*cRLIssuer*» издателем. Когда в рамках URI-схемы используются HTTP- или FTP-протоколы, то URI-идентификатор должен, в обязательном порядке, быть указателем на одиночный СОС, закодированный в соответствие с DER-правилами (RFC-2585). Когда прикладные системы реализованы на основе HTTP-сервера и получают доступ к нему по URI-идентификатору, тогда целесообразно, чтобы в поле «*content-type*» заголовка ответного сообщения был указан тип среды доставки «*application/pkix-crl*». Когда прикладные системы используют LDAP-протокол для доступа к LDAP-серверу по URI-идентификатору (RFC-4516), тогда URI-идентификатор должен, в обязательном порядке, включать поле «<*dn*>», содержащее уникальное имя держателя СОС, он должен, в обязательном порядке, включать одиночное поле «<*attrdesc*>», содержащее описание соответствующего атрибута, который указывает на держателя СОС (RFC-4523), и целесообразно, чтобы он включал поле «<*host*>» (например, <*ldap://ldap.example.com/cn=example%20CA,dc=example,dc=com?certificateRevocationList; binary*>). Пропуск поля «<*host*>» (например, <*ldap:///cn=CA, dc=example,dc=com?authorityRevocationList;binary*>) может быть эффективным тогда, когда клиент обладает какими-нибудь априорными знаниями для соединения с соответствующим сервером. Если поле «<*host*>» представлено, то целесообразно, чтобы последовательность «*DistributionPointName*» включала, по крайней мере, один URI-идентификатор для LDAP- или HTTP-сервера.

Если последовательность «*DistributionPointName*» в субполе «*cRLDistributionPoints*» включает одиночное значение «*nameRelativeToCRLIssuer*», то оно является фрагментом уникального имени. Данный фрагмент присоединяется к уникальному имени издателя СОС (в формате, определённом в Рекомендации ITU-T X.500), которое предназначено для получения наименования точки распространения сертификатов. Если последовательность «*DistributionPointName*» содержит субпоследовательность «*cRLIssuer*», то фрагмент наименования присоединяется к уникальному имени, которое содержит эта субпоследовательность. В противном случае, фрагмент имени присоединяется к уникальному имени издателя сертификата. Целесообразно, чтобы УЦ, придерживающиеся данного стандарта, не использовали значение «*nameRelativeToCRLIssuer*» для указания наименований точек распространения сертификатов. Последовательность «*DistributionPointName*» в субполе «*cRLDistributionPoints*» не должна использоваться для альтернативных значений «*nameRelativeToCRLIssuer*», если субпоследовательность «*cRLIssuer*» содержит более одного уникального имени.

Если последовательность «*DistributionPoint*» не содержит субпоследовательности «*reasons*», то СОС должен обязательно включать все причины аннулирования сертификатов. Данный стандарт не рекомендует сегментацию СОС с помощью кода причины («*reasons*»). Если УЦ, придерживающиеся данного стандарта, включают в сертификат субполе «*cRLDistributionPoints*», то данное субполе должно, в обязательном порядке, включать, по крайней мере, одну последовательность «*DistributionPoint*», которая указывает на СОС, содержащий сертификат со всеми причинами.

Субпоследовательность «*cRLIssuer*» идентифицирует субъект, который подписал и издал СОС. Если такая субпоследовательность представлена, то она должна, в обязательном порядке, включать только уникальное имя СОС, скопированное из поля «*Issuer*», на которое указывает последовательность «*DistributionPoint*». Кодировка имени в субпоследовательности «*cRLIssuer*» должно точно совпадать с кодировкой в поле «*Issuer*» СОС. Если субпоследовательность «*cRLIssuer*» включена, а уникальное имя в этой субпоследовательности

не соответствует СЕК-записи, закодированной по правилам, представленным в Рекомендации ITU-T X.500, или по правилам LDAP-протокола, и указывающей место расположения СОС, то УЦ, придерживающиеся данного стандарта, обязаны включать в сертификат субпоследовательность «*distributionPoint*».

```
id-ce-cRLDistributionPoints          OBJECT IDENTIFIER ::= { id-ce 31 }

CRLDistributionPoints ::=            SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::=                SEQUENCE {
        distributionPoint                [0]   DistributionPointName OPTIONAL,
        reasons                          [1]   ReasonFlags OPTIONAL,
        cRLIssuer                        [2]   GeneralNames OPTIONAL }

DistributionPointName ::=            CHOICE {
        fullName                         [0]   GeneralNames,
        nameRelativeToCRLIssuer          [1]   RelativeDistinguishedName }

ReasonFlags ::=                      BIT STRING {
        unused                           (0),
        keyCompromise                    (1),
        cACompromise                     (2),
        affiliationChanged               (3),
        superseded                       (4),
        cessationOfOperation             (5),
        certificateHold                  (6),
        privilegeWithdrawn               (7),
        aACompromise                     (8) }
```

§4.2.1.14 Запрет на использование субполя «anyPolicy»

Субполе «*inhibitAnyPolicy*» может использоваться в сертификатах, которые выпускаются для УЦ. Это субполе указывает, что OID некоторой политики, со значением { *2 5 29 32 0* }, не рассматривается при точном сравнении с другими политиками сертификации, за исключением случая, когда он представлен в промежуточном само-изданном сертификате УЦ. Содержащее в данном субполе значение определяет число дополнительных не само-изданных сертификатов, которые могут присутствовать на МС, пока политика, указанная в субполе «*anyPolicy*», не будет больше разрешена. Например, значение в этом субполе указывает на то, что субполе «*anyPolicy*» может обрабатываться в сертификатах, изданных владельцем данного сертификата, но не в дополнительных сертификатах, представленных на МС.

УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «критичное».

```
id-ce-inhibitAnyPolicy          OBJECT IDENTIFIER ::=       { id-ce 54 }
InhibitAnyPolicy ::=            SkipCerts
SkipCerts ::=                   INTEGER (0..MAX)
```

§4.2.1.15 Самый последний COC (известный также как «точка распространения обновлённого (*delta*) COC»)

Субполе «*freshestCRL*» указывает на то, как была получена информация об обновлённом COC. УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «не критичное». Проблемы обеспечения COC будут рассмотрены ниже.

Для субполя «*freshestCRL*» используется аналогичный синтаксис, как и для субполя «*cRLDistributionPoints*» (§4.2.1.13). Одни и те же правила применяются к обеим субполям.

```
id-ce-freshestCRL       OBJECT IDENTIFIER ::=       { id-ce 46 }
FreshestCRL ::=         CRLDistributionPoints
```

§4.2.2 Частные расширения для Интернет/PKI-инфраструктуры

Далее рассматриваются два расширения (два субполя в поле «Расширения» сертификата), используемые в рамках Интернет/PKI-инфраструктуры. Эти субполя могут использовать непосредственно прикладными системами при интерактивном получении информации об издателе или владельце сертификата. Каждое субполе включает последовательность данных, состоящую из способов доступа и точках доступа. Способ доступа представляет собой объектный идентификатор, который указывает на тип доступной информации. Точка доступа — это последовательность «*GeneralName*» (обобщённое наименование), которая косвенным образом указывает на место размещения и способ получения информации.

Для частных расширений, в рамках Интернет/PKI-инфраструктуры, определены объектные идентификаторы. Объектные идентификаторы, связанные с

частными расширениями, образуют отдельную ветвь «*id-pe*» в иерархическом дереве идентификаторов «*id-pkix*». Любое другое расширение в рамках Интернет/PKI-инфраструктуры будет также входить в ветвь «*id-pe*».

```
id-pkix     OBJECT IDENTIFIER  ::=     { iso(1) identified-organization(3) dod(6) internet(1)
                                        security(5) mechanisms(5) pkix(7) }

  id-pe     OBJECT IDENTIFIER  ::=     { id-pkix 1 }
```

§4.2.2.1 Доступ к информации УЦ

Субполе «*authorityInfoAccess*» указывает на то, как получить доступ к информации и услугам издателя сертификата, в котором представлено это субполе. Информация и услуги могут включать интерактивные услуги по подтверждению подлинности и данные о политике УЦ. (Место расположения СОС в данном субполе не указывается. Такая информация содержится в субполе «*cRLDistributionPoints*».) Данное субполе может входить в состав сертификатов конечных пользователей или УЦ. УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «не критичное».

```
id-pe-authorityInfoAccess           OBJECT IDENTIFIER ::= { id-pe 1 }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=  SEQUENCE {
        accessMethod        OBJECT IDENTIFIER,
        accessLocation      GeneralName  }

id-ad                               OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-caIssuers                     OBJECT IDENTIFIER ::= { id-ad 2 }

id-ad-ocsp                          OBJECT IDENTIFIER ::= { id-ad 1 }
```

Каждая запись (субпоследовательность) в последовательности «*AuthorityInfoAccessSyntax*» определяет формат и точку доступа к дополнительной информации, предоставляемой издателем сертификата, в котором представлено субполе «*authorityInfoAccess*». Тип и формат информации указываются с помощью субпоследовательности «*accessMethod*». Субпоследовательность «*accessLocation*» место размещения (точку доступа к) информации. Способ извлечения

данных может быть определён на основании информации, содержащейся в субпоследовательности «*accessMethod*» или «*accessLocation*».

Данный стандарт устанавливает два идентификатора для способов доступа: «*id-ad-caIssuers*» и «*id-ad-ocsp*».

В СЕРТ|ОК, идентификатор «*id-ad-caIssuers*» используется тогда, когда дополнительная информация содержит перечень сертификатов, которые были изданы для УЦ, который, в свою очередь, издал сертификат, включающий это субполе «*authorityInfoAccess*». Описание со справочными данными, изданное УЦ, предназначено для помощи пользователям сертификата при выборе МС, заканчивающегося в точке, которой доверяет пользователь сертификата.

Если идентификатор «*id-ad-caIssuers*» представлен как способ доступа (субпоследовательность «*accessMethod*»), то субпоследовательность «*accessLocation*» указывает на сервер, в котором храниться описание со справочными данными, и протокол доступа с целью получения этого описания. Субпоследовательность «*accessLocation*» представляет собой обобщённое наименование («*GeneralName*»), которое может иметь несколько форматов.

Если субпоследовательность «*accessLocation*» является наименованием в формате «*directoryName*», то прикладная система должна получать необходимую информацию из любого СЕК-сервера, который настраивается исходя из локальных условий. Запись в формате «*directoryName*» содержит сертификаты УЦ в атрибутах «*crossCertificatePair*» и/или «*cACertificate*», как это определено в стандарте RFC-4523. Протокол, используемый прикладной системой (прикладным процессом) для доступа к СЕК-сегменту (например, DAP- или LDAP-протокол), выбирается исходя из локальных условий.

Когда доступ к информации обеспечивается с помощью LDAP-протокола, тогда целесообразно, чтобы субпоследовательность «*accessLocation*» представляла собой URI-идентификатор («*uniformResourceIdentifier*»). URI-идентификатор LDAP-сервера (RFC-4516) должен, в обязательном порядке, включать поле «<*dn*>», содержащее уникальное имя владельца сертификатов, он должен, в обязательном порядке, включать субпоследовательность «<*attributes*>», содер-

жащая список описаний атрибутов, которые содержат сертификаты или пары кросс-сертификатов в DER-коде (RFC-4523), и целесообразно, чтобы он включал поле «<*host*>» (например, <*ldap://ldap.example.com/cn=CA,dc=example, dc=com? cACertificate; binary*>). Пропуск поля «<*host*>» (например, <*ldap:/// cn=exampleCA,dc=example,dc=com?cACertificate;binary*>) может быть эффективным тогда, когда клиент обладает какими-нибудь априорными знаниями для соединения с соответствующим сервером.

Когда доступ к информации обеспечивается с помощью HTTP- или FTP-протокола, тогда субпоследовательность «*accessLocation*» должна быть, в обязательном порядке, URI-идентификатором («*uniformResourceIdentifier*»). В этом случае, сам URI-идентификатор должен быть, в обязательном порядке, указателем на одиночный сертификат, закодированный по DER-правилам (RFC-2585), или на совокупность сертификатов, расположенных в CMS-сообщении (*Cryptographic Message Syntax*, «*certs-only*») и закодированных по BER- или DER-правилам (RFC-2797).

Прикладные системы, придерживающиеся данного стандарта и использующие HTTP- или FTP-протокол для обеспечения доступа к сертификатам, обязаны предоставлять доступ к индивидуальным сертификатам, закодированным по DER-правилам, а также целесообразно, чтобы такие прикладные системы обеспечивали обмен CMS-сообщениями («*certs-only*»).

Целесообразно, чтобы прикладные системы, функционирующие с использованием URI-идентификаторов при обеспечении доступа к HTTP-серверу, указывали тип доставки «*application/pkix-cert*» в поле заголовка «*content-type*» ответного сообщения на запрос одиночного сертификата, закодированного по DER-правилам. Кроме того, целесообразно, чтобы такие прикладные системы указывали тип доставки «*application/pkcs7-mime*» (RFC-2797) в поле заголовка «*content-type*» ответного сообщения при обмене CMS-сообщениями («*certs-only*»). При использовании FTP-протокола, целесообразно, чтобы имя файла, содержащего одиночный сертификат, закодированный по DER-правилам, включало суффикс «.*cer*» (RFC-2585), а имя файла, содержащего CMS-сообще-

ние («*certs-only*»), — «*.p7c*» (RFC-2797). Касаясь клиентов, то они могут использовать расширение «тип доставки» («*media type*») или «файл» («*file*») как указатель на содержание, но, в то же время, не целесообразно, чтобы оно зависело только от наличия корректного расширения «тип доставки» или «файл» в ответе сервера.

Семантика других форматов имён «*id-ad-caIssuers*» и «*accessLocation*» в данном стандарте не рассматривается.

Субполе «*authorityInfoAccess*» может включать несколько идентификаторов «*id-ad-caIssuers*» и субпоследовательностей «*accessMethod*». Различные идентификаторы и способы доступа могут указывать, либо на различные способы обеспечения доступа к одной и той же информации, либо на различную информацию. Когда используются идентификаторы «*id-ad-caIssuers*» и субпоследовательности «*accessMethod*», тогда целесообразно, чтобы, по крайней мере, один идентификатор указывал на точку доступа («*accessLocation*»), которая использовала бы URI-идентификаторы при обеспечении доступа к HTTP- или LDAP-серверу (RFC-2616 и RFC-4516).

Идентификатор «*id-ad-ocsp*» используется в тех случаях, когда информация об аннулированных сертификатов, содержащих данное расширение, доступна с помощью интерактивного протокола проверки состояния сертификата (OCSP, RFC-6960).

Когда идентификатор «*id-ad-ocsp*» используется в качестве указателя на способ доступа (субпоследовательность «*accessMethod*»), тогда субпоследовательность «*accessLocation*» определяет место расположение сетевого субъекта, использующего OCSP-протокол в соответствие с RFC-6960.

Дополнительные определители типов доступа могут быть представлены в других стандартах Интернет/PKI-инфраструктуры.

§4.2.2.2 Доступ к информации владельца сертификата

Это субполе «*subjectInfoAccess*» указывает на то, как получить доступ владельцу сертификата, содержащего это расширение, к информации и услу-

гам. Когда держателем сертификата является УЦ, тогда информация и услуги могут включать интерактивные услуги по подтверждению подлинности и данные о политике УЦ. Когда держателем сертификата является конечные пользователь, тогда информация определяет тип предлагаемых услуг и как получить доступ к ним. В таком случае, содержание субполя «*subjectInfoAccess*» определено в стандартном протоколе для обеспечивающих служб. Это субполе может входить в состав сертификатов конечного пользователя или УЦ. УЦ, придерживающиеся данного стандарта, обязаны помечать данное субполе как «не критичное».

```
id-pe-subjectInfoAccess          OBJECT IDENTIFIER ::= { id-pe 11 }

SubjectInfoAccessSyntax  ::=     SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription  ::=           SEQUENCE {
        accessMethod                     OBJECT IDENTIFIER,
        accessLocation                   GeneralName  }
```

Каждая запись в последовательности «*SubjectInfoAccessSyntax*» указывает на формат и место расположения дополнительной информации, предоставляемой владельцем сертификата, в котором содержится это субполе. Тип и формат информации определяется с помощью субпоследовательности «*accessMethod*», место расположения информации определяется с помощью субпоследовательности «*accessLocation*». Содержание субпоследовательности «*accessMethod*» или специализированное содержание субпоследовательности «*accessLocation*» подразумевают способ получения информации.

Данный стандарт рассматривает один способ доступа, используемый тогда, когда владельцем сертификата является УЦ, и один способ доступа, используемый тогда, когда владельцем сертификата является конечный пользователь. Дополнительные способы доступа могут быть п=описаны в дальнейшем в стандартных протоколах других сетевых служб.

Объектный идентификатор «*id-ad-caRepository*» используется тогда, когда владельцем сертификата является УЦ, которые публикует сертификаты и размещает их в репозитарии. Субпоследовательность «*accessLocation*» рассмат-

ривается как обобщённое имя «*GeneralName*», которое может иметь несколько форматов.

Если субпоследовательность «*accessLocation*» является наименованием в формате «*directoryName*», то прикладная система должна получать необходимую информацию из любого СЕК-сервера, который настраивается исходя из локальных условий. Когда субполе «*subjectInfoAccess*» используется в указателе на сертификаты УЦ, тогда запись в формате «*directoryName*» содержит сертификаты УЦ в атрибутах «*crossCertificatePair*» и/или «*cACertificate*», как это определено в стандарте RFC-4523. Протокол, используемый прикладной системой (прикладным процессом) для доступа к СЕК-сегменту (например, DAP- или LDAP-протокол), выбирается исходя из локальных условий.

Когда доступ к информации обеспечивается с помощью LDAP-протокола, тогда целесообразно, чтобы субпоследовательность «*accessLocation*» представляла собой URI-идентификатор («*uniformResourceIdentifier*»). URI-идентификатор LDAP-сервера (RFC-4516) должен, в обязательном порядке, включать поле «<*dn*>», содержащее уникальное имя владельца сертификатов, он должен, в обязательном порядке, включать субпоследовательность «<*attributes*>», содержащая список описаний атрибутов, которые содержат сертификаты или пары кросс-сертификатов в DER-коде (RFC-4523), и целесообразно, чтобы он включал поле «<*host*>» (например, <*ldap://ldap.example.com/cn=CA,dc=example, dc=com? cACertificate; binary, crossCertificatePair;binary* >). Пропуск поля «<*host*>» (например, <*ldap://cn=exampleCA,dc=example,dc=com?cACertificate; binary*>) может быть эффективным тогда, когда клиент обладает какими-нибудь априорными знаниями для соединения с соответствующим сервером.

Когда доступ к информации обеспечивается с помощью HTTP- или FTP-протокола, тогда субпоследовательность «*accessLocation*» должна быть, в обязательном порядке, URI-идентификатором («*uniformResourceIdentifier*»). В этом случае, сам URI-идентификатор должен быть, в обязательном порядке, указателем, либо на одиночный сертификат, закодированный по DER-правилам (RFC-

2585), либо на совокупность сертификатов, расположенных в CMS-сообщении («*certs-only*») и закодированных по BER- или DER-правилам (RFC-2797).

Прикладные системы, придерживающиеся данного стандарта и использующие HTTP- или FTP-протокол для обеспечения доступа к сертификатам, обязаны предоставлять доступ к индивидуальным сертификатам, закодированным по DER-правилам, а также целесообразно, чтобы такие прикладные системы обеспечивали обмен CMS-сообщениями («*certs-only*»).

Целесообразно, чтобы прикладные системы, функционирующие с использованием URI-идентификаторов при обеспечении доступа к HTTP-серверу, указывали тип доставки «*application/pkix-cert*» (RFC-2585) в поле заголовка «*content-type*» ответного сообщения на запрос одиночного сертификата, закодированного по DER-правилам. Кроме того, целесообразно, чтобы такие прикладные системы указывали тип доставки «*application/pkcs7-mime*» (RFC-2797) в поле заголовка «*content-type*» ответного сообщения при обмене CMS-сообщениями («*certs-only*»). При использовании FTP-протокола, целесообразно, чтобы имя файла, содержащего одиночный сертификат, закодированный по DER-правилам, включало суффикс «*.cer*» (RFC-2585), а имя файла, содержащего CMS-сообщение («*certs-only*»), — «*.p7c*» (RFC-2797). Касаясь клиентов, то они могут использовать расширение «тип доставки» («*media type*») или «файл» («*file*») как указатель на содержание, но, в то же время, не целесообразно, чтобы оно зависело только от наличия корректного расширения «тип доставки» или «файл» в ответе сервера.

Семантика других форматов имён «*id-ad-caRepository*» и «*accessLocation*» в данном стандарте не рассматривается.

Субполе «*subjectInfoAccess*» может включать несколько идентификаторов «*id-ad-caRepository*» и субпоследовательностей «*accessMethod*». Различные идентификаторы и способы доступа могут указывать, либо на различные способы обеспечения доступа к одной и той же информации, либо на различную информацию. Когда используются идентификаторы «*id-ad-caRepository*» и субпоследовательности «*accessMethod*», тогда целесообразно, чтобы, по крайней

мере, один идентификатор указывал на точку доступа («*accessLocation*»), которая использовала бы URI-идентификаторы при обеспечении доступа к HTTP- или LDAP-серверу (RFC-2616 и RFC-4516).

Когда владелец сертификата предоставляет услуги по применению меток времени (*timestamping services*) на основе соответствующего протокола (*Time Stamp Protocol*, RFC-3161), тогда используется OID «*id-ad-timeStamping*». Там, где услуги по применению меток времени предоставляются на основе HTTP- или FTP-протокола, субпоследовательность «*accessLocation*» должна быть, в обязательном порядке, URI-идентификатором («*uniformResourceIdentifier*»). Там, где услуги по применению меток времени предоставляются на основе службы электронной почты, субпоследовательность «*accessLocation*» должна быть представлена, в обязательном порядке, в формате «*rfc822Name*». Если услуги по применению меток времени предоставляются на основе протоколов транспортного и сетевого уровней Интернет-архитектуры, то могут использоваться два формата имён «*dNSName*» или «*iPAddress*». Семантика других форматов имён в субпоследовательности «*accessLocation*» (когда субпоследовательность «*accessMethod*» является OID «*id-ad-timeStamping*») в данном стандарте не рассматривается.

Дополнительные определители типов доступа могут быть представлены в других стандартах Интернет/PKI-инфраструктуры.

```
id-ad                   OBJECT IDENTIFIER ::= { id-pkix 48 }
id-ad-caRepository      OBJECT IDENTIFIER ::= { id-ad 5 }
id-ad-timeStamping      OBJECT IDENTIFIER ::= { id-ad 3 }
```

V ОПИСАНИЕ COC И СУБПОЛЯ «РАСШИРЕНИЯ COC»

Из рассмотренного выше следует, что одной из целей описания COC в Рекомендации ITU-T X.509 2-ой версии является обеспечение функциональной совместимости в рамках Интернет/PKI-инфраструктуры и её долговременного использования. Для достижения указанной цели, далее рассматриваются принципы и правила использования субполя «Расширения COC», а также приводят-

ся некоторые предположения относительно «природы» информации, содержащейся в СОС.

СОС могут использоваться во многих прикладных и информационно-технологических системах, охватывая широкий спектр проблем обеспечения функциональной совместимости, и ещё больший спектр требований по надёжному и корректному функционированию. Этот стандарт:

✓ устанавливает единую основу для универсальных прикладных систем и открытых ИТС, которые требуют всесторонней функциональной совместимости;

✓ определяет набор данных, которые могут быть представлены в каждом СОС;

✓ определяет единые места в СОС, в которых располагаются наиболее часто используемые атрибуты, а также единые формы представления для таких атрибутов.

СОС выпускаются издателями СОС. Издатель СОС представляет собой, либо УЦ, либо субъект, который авторизован УЦ для выпуска сертификатов. УЦ публикуют СОС с целью предоставления информации о состоянии (статусе) сертификатов, которые они издали. Тем не менее, УЦ могут делегировать эту функцию (ответственность за реализацию этой функции) другому доверенному центру (субъекту).

Каждый СОС имеет конкретную область применения. Область применения СОС представляет собой совокупность сертификатов, которые могли быть размещены в конкретном СОС. Например, областью применения могли бы быть:

▪ «все сертификаты, изданные УЦ *X*», «все сертификаты УЦ, изданные УЦ *X*»;

▪ «все сертификаты, изданные УЦ *X*, которые были удалены по причинам компрометации ключа и компрометации УЦ»;

▪ совокупность сертификатов, основанных на любой локальной информации, такой как «все сертификаты, изданные для служащих Национального

института стандартов и технологий, работающих в Боулдер (штат Колорадо, США)[°]».

Заполняемый COC (*complete CRL*) содержит список всех не просроченных сертификатов, в границах своей области применения, которые были аннулированы (отозваны) по тем причинам, указанным в описании их области применения. *Заполненный* COC (*full and complete CRL*) содержит список всех не просроченных сертификатов, изданных УЦ, которые были аннулированы по любым причинам. (Примечание. Так как УЦ и издатели УЦ идентифицируются по наименованию, область применения COC не касается ключа, который используются для подписи COC, или ключа(ей), который(е) используется(ются) для подписи сертификатов.)

Если область применения COC включает один или несколько сертификатов, изданных субъектом, который не является издателем COC, то она является *косвенным* COC. Область применения косвенного COC может быть ограничена сертификатами, изданными одним УЦ, или может включать сертификаты, изданные несколькими УЦ. Если издателем косвенного COC является УЦ, то область применения косвенного COC может также включать сертификаты, выпущенные издателем COC.

Издатель COC также может формировать *усечённые* COC (*delta CRLs*). Усечённые COC содержат списки только тех сертификатов, в границах их областей применения, у которых состояние аннулирования изменилось с момента издания основного заполняемого COC. Основной заполняемый COC называется *базовым* COC. Область применения усечённого COC может совпадать с областью применения базового COC, на который он ссылается.

Данный стандарт рассматривает одно частное расширение в COC для Интернет/PKI-инфраструктуры, но не устанавливает ни одного частного расширения для записей в COC.

[°] «All certificates issued to the NIST employees located in Boulder».

Прикладные системы и ИТС, предъявляющие дополнительные или специализированные требования, могут создаваться на основе данного стандарта или могут его заменить.

УЦ, придерживающиеся данного стандарта, могут не требовать издания СОС, если обеспечиваются иные способы аннулирования или определения состояния сертификатов. Когда СОС издаются, тогда они должны быть, в обязательном порядке, второй версии, и включать:

✓ данные, указывающие на дату выпуска следующего СОС, в субполе «*nextUpdate*»;

✓ последовательность «*cRLNumber*» в субполе «Расширения СОС»;

✓ последовательность «*authorityKeyIdentifier*» в субполе «Расширения СОС».

Прикладные системы, придерживающиеся данного стандарта, и которые обслуживают СОС, необходимые для обработки первой и второй версий сертификатов, заполняют СОС с целью предоставления информации по отзыву всех сертификатов, изданных одним УЦ. Нет необходимости задействовать прикладные системы, придерживающиеся данного стандарта, для обработки усечённых СОС, косвенных СОС и СОС, у которых в области применения не входит ни один сертификат, изданный одним УЦ.

§5.1 Поля СОС

Далее представлен синтаксис СОС в соответствие с Рекомендацией ITU-T X.509 2-ой версии. Для вычисления подписи, данные, которые должны быть подписаны, представляются ASN.1-коде по DER-правилам. ASN.1-кодирование по DER-правилам представляет собой кодовую систему, в которой для каждого элемента данных используются указатель (*tag*), длина последовательности символов (*length*) и само символьное значение (*value*).

```
CertificateList  ::=          SEQUENCE  {
        tbsCertList                     TBSCertList,
        signatureAlgorithm              AlgorithmIdentifier,
        signatureValue                  BIT STRING  }
```

```
TBSCertList  ::=          SEQUENCE  {
        version           Version OPTIONAL,
                                  -- если представлено, то должно быть 2-й версии
        Signature                 AlgorithmIdentifier,
        Issuer                    Name,
        thisUpdate                Time,
        nextUpdate                Time OPTIONAL,
        revokedCertificates       SEQUENCE OF SEQUENCE  {
                userCertificate           CertificateSerialNumber,
                revocationDate            Time,
                crlEntryExtensions        Extensions OPTIONAL
                                  -- если представлено, то должно быть 2-й версии
                                  }  OPTIONAL,
        crlExtensions             [0]     EXPLICIT Extensions OPTIONAL
                                  -- если представлено, то должно быть 2-й версии
}
```

Поля «*Version*», «*Time*», «*CertificateSerialNumber*» и «Расширения» представлены в §4.1 в ASN.1-коде, а поле «*AlgorithmIdentifier*» представлено в §4.1.1.2.

Далее описывается применение СОС в соответствие с Рекомендацией ITU-T X.509 2-ой версии в Интернет/PKI-инфраструктуре.

§5.1.1 Субполя (состав) поля «CertificateList»

Поле «*CertificateList*» представляет собой последовательность символов, состоящей из трёх субполей, описание которых рассматривается далее.

§5.1.1.1 Субполе «tbsCertList»

Это субполе является первым в перечне сертификатов и включает имя издателя, дату выпуска, дату выпуска следующего СОС, дополнительный список отозванных сертификатов и дополнительные субполя в поле «Расширения» СОС. Когда отозванных сертификатов нет, то перечень отозванных сертификатов отсутствует. Если один или несколько сертификатов отозвано, то каждая запись в списке отозванных сертификатов включает серийный номер сертификата пользователя, дату отзыва и дополнительную запись в поле «Расширения» СОС.

§5.1.1.2 Субполе «signatureAlgorithm»

Это субполе содержит идентификатор алгоритма «*AlgorithmIdentifier*», используемого издателем СОС для подписи поля «*CertificateList*». Совокупность алгоритмов и их идентификаторов представлена в §4.1.1.2 (RFC-3279, RFC-4055 и RFC4491). Тем не менее, могут использоваться и другие алгоритмы формирования подписи.

Это субполе должно, в обязательном порядке, включать точно такой же идентификатор алгоритма, который указан в субполе «*Signature*» поля «*tbsCertList*» (§5.1.2.2).

§5.1.1.3 Субполе «signatureValue»

Это субполе содержит цифровую подпись, которая была вычислена по последовательности символов в ASN.1/DER-коде, содержащейся в поле «*tbsCertList*». Последовательность символов в ASN.1/DER-коде, содержащаяся в поле «*tbsCertList*», представляет собой входные данные функции вычисления подписи. Это значение подписи закодировано в формате битовой последовательности («*BIT STRING*»), и включено в субполе «*signatureValue*». Детали этой процедуры представлены в стандартах RFC-3279, RFC-4055 и RFC-4491.

УЦ, которые также выпускают СОС, могут использовать один закрытый ключ или разные закрытые ключи для цифровой подписи сертификатов и СОС. Когда применяются разные закрытые ключи, тогда каждый из открытых ключей, связанных с этими закрытыми ключами, указывается в отдельном сертификате. В одном сертификате в субполе «*Key usage*» поля «*Расширения*» установлен флаг «*keyCertSign*», а в другом сертификате в субполе «*Key usage*» поля «*Расширения*» установлен флаг «*cRLSign*» (§4.2.1.3). Если используются разные закрытые ключи, то сертификаты, издаваемые УЦ, содержат идентификатор первого ключа УЦ, а соответствующие СОС содержат идентификатор второго ключа УЦ. Использование отдельных сертификатов УЦ при подтверждении подлинности подписей сертификатов и подписей СОС может обеспечить лучшие показатели защищённости. Тем не менее, такой подход влечёт за собой дополнительные расходы для прикладных систем, а также он может ограничить

функциональную совместимость. Многие прикладные системы формируют МС, а затем подтверждают подлинность такого маршрута. Проверка СОС, в свою очередь, требует отдельного МС, который должен формироваться и проверяться на предмет его подлинности относительно сертификата УЦ, необходимого для подтверждения подлинности подписи в СОС. Прикладные системы, которые осуществляют проверку СОС, обязаны проводить ПрПП МС, если сертификаты и СОС включают цифровые подписи, сформированные с помощью одного и того же закрытого ключа УЦ. Целесообразно, чтобы такие прикладные системы проводили ПрПП МС, если сертификаты и СОС включают цифровые подписи, сформированные с помощью разных закрытых ключей УЦ.

§5.1.2 Список сертификатов, подлежащих подписанию

Поле «*TBSCertList*» (список сертификатов), которое должно быть подписано, представляет собой последовательность обязательных и дополнительных субполе. Обязательные субполя указывают на издателя СОС, используемы алгоритм подписания СОС и дату и время выпуска СОС.

Дополнительные (не обязательные) поля включают дату и время, когда издатель СОС выпустит следующий СОС, списки отозванных сертификатов и субполя в поле «Расширения» СОС. Перечень отозванных сертификатов является не обязательным в том случае, когда УЦ не аннулирует любые просроченные сертификаты, которые он издал. Этот стандарт требует, чтобы издатели СОС, придерживающиеся данного стандарта, включали в издаваемые ими СОС последовательности «*nextUpdate*», «*cRLNumber*» и «*authorityKeyIdentifier*» в рамках субполя «Расширения СОС».

§5.1.2.1 Версия СОС

Это не обязательное субполе указывает на версию закодированного СОС. Когда поле «Расширения» используется, как требует данный стандарт, тогда данное субполе должно быть представлено в обязательном порядке, и оно

должно, в обязательном порядке, указывать на вторую версию (целое число «*1*»).

§5.1.2.2 Подпись

Это субполе содержит идентификатор алгоритма, используемого при подписании COC. Стандарты RFC-3279, RFC-4055 и RFC-4491 содержат идентификаторы наиболее популярных алгоритмов ЭЦП, используемых в Интернет/PKI-инфраструктуре.

В этом субполе, в обязательном порядке, должен содержаться такой же идентификатор алгоритма, как и в субполе «*signatureAlgorithm*» поля «*CertificateList*» (§5.1.1.2).

§5.1.2.3 Имя (наименование) издателя COC

Это субполе «*Issuer*» идентифицирует субъекта, который подписал и выпустил COC. Параметр подлинности издателя доставляется в субполе «*Issuer*». Альтернативное имя различных форматов может быть также включено в субполе «*issuerAltName*» поля «Расширения» COC. В субполе «*Issuer*» должно, в обязательном порядке, содержаться не пустое уникальное имя в соответствие с Рекомендацией ITU-T X.500. Субполе «*Issuer*» соответствует типу «*Name*», как это определено в Рекомендации ITU-T X.501. Содержание этого поля должно, в обязательном порядке, кодироваться по тем правилам, какие определены для поля «*Issuer*» в сертификате (§4.1.2.4).

§5.1.2.4 Дата издания COC

Это субполе «*thisUpdate*» содержит дату выпуска данного COC. Субполе «*thisUpdate*» может кодироваться как UTCTime- или GeneralizedTime-время.

Издатели COC, придерживающиеся данного стандарта, обязаны кодировать даты, размещаемые в субполе «*thisUpdate*», как:

- UTCTime-время вплоть до 2049 года (включительно);
- GeneralizedTime-время, начиная с 2050 года и далее.

Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны обрабатывать даты, закодированные как UTCTime- или GeneralizedTime-время.

Если используется UTCTime-кодировка, то субполе «*thisUpdate*» должно, в обязательном порядке, интерпретироваться так, как указано в §4.1.2.5.1, а если используется GeneralizedTime-кодировка — в §4.1.2.5.2.

§5.1.2.5 Дата издания следующего COC

Это субполе «*nextUpdate*» содержит дату выпуска следующего COC. Следующий COC может бы быть издан до указанной даты, но не будет издан позже этой даты. Целесообразно, чтобы издатели COC выпускали их с субполем «*nextUpdate*», в котором бы указывалось время равное или более позднее, чем время, указанное во всех предшествующих COC. Субполе «*nextUpdate*» может кодироваться как UTCTime- или GeneralizedTime-время.

Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны включать субполе «*nextUpdate*» во все COC. (Примечание. ASN.1-синтаксис поля «*TBSCertList*» определяет это субполе как не обязательное, что соответствует ASN.1-конструкции, которая представлена в Рекомендации ITU-T X.509.) Данный стандарт не рассматривает процедурную характеристику и её алгоритм для клиентов, которые обрабатывают COC, не содержащие субполе «*nextUpdate*».

Издатели COC, придерживающиеся данного стандарта, обязаны кодировать даты, размещаемые в субполе «*thisUpdate*», как:

- UTCTime-время вплоть до 2049 года (включительно);
- GeneralizedTime-время, начиная с 2050 года и далее.

Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны обрабатывать даты, закодированные как UTCTime- или GeneralizedTime-время.

Если используется UTCTime-кодировка, то субполе «*thisUpdate*» должно, в обязательном порядке, интерпретироваться так, как указано в §4.1.2.5.1, а если используется GeneralizedTime-кодировка — в §4.1.2.5.2.

§5.1.2.6 Аннулированные сертификаты

Когда отозванных сертификатов нет, тогда список аннулированных сертификатов должен, в обязательном порядке, отсутствовать. В противном случае, отозванные сертификаты перечисляются по их последовательным номерам. Сертификаты, отозванные УЦ, однозначно определяются по их последовательному номеру. Далее описывается формат даты, когда произошло аннулирование. Формат времени, указанного в последовательности «*revocationDate*» субполя «*revokedCertificates*», должен, в обязательном порядке, соответствовать формату, представленному в §5.1.2.4. Дополнительная информация может быть размещена в расширениях записей СОС, которые рассматриваются далее.

§5.1.2.7 Субполе «Расширения СОС»

Субполе «*crlExtensions*» может быть представлено только в СОС 2-ой версии (§5.1.2.1). Если это поле присутствует, то оно имеет форму, состоящую из нескольких последовательностей (рассматривается далее).

§5.2 Субполе «Расширения СОС»

Эти расширения СОС определены в стандартах ANSI X9, ISO/IEC и ITU-T X.509 для 2-ой версии СОС и определяют способы привязки дополнительных атрибутов с СОС. Формат СОС в соответствие с Рекомендацией ITU-T X.509 2-ой версии также разрешает различным PKI-инфраструктурам устанавливать частные расширения для доставки информации, являющейся уникальной этих инфраструктур. Каждое расширение в СОС может быть помечено как «критичное» или «не критичное». Если СОС содержит критичное расширение, которое не может обработать прикладной процесс, то этот прикладной процесс обязан не использовать этот СОС для определения статуса сертификатов. Однако, прикладные системы могут игнорировать не распознаваемые не критичные расширения СОС. Различные PKI-инфраструктуры могут сами выбирать, необходимо ли им включать в СОС расширения, которые не представлены в данном стандарте. Тем не менее, целесообразно «с особой осторожностью» включать с СОС

какие-либо критичные расширения, которые могли бы использоваться в общем контексте.

Издателям СОС, придерживающимся данного стандарта, рекомендуется включать во все издаваемые ими СОС субполе «*crlExtensions*» с последовательностями «*authorityKeyIdentifier*» и «*cRLNumber*».

§5.2.1 Последовательность «authorityKeyIdentifier»

Последовательность «*authorityKeyIdentifier*» (идентификатор ключа УЦ) указывает на средства идентификации соответствующего закрытого ключа, используемого при подписании СОС. Процедура идентификации может основываться, либо идентификаторе ключа (идентификатор ключа владельца, указанный в сертификате для подписания СОС), либо имя издателя и последовательный номер. Эта последовательность наиболее полезна там, где издатель имеет несколько ключей для подписи, либо вследствие нескольких одновременно действующих пар криптоключей, либо вследствие технологической перенастройки.

Издатели СОС, придерживающиеся данного стандарта, обязаны использовать способ идентификации криптоключей, а также — включать эту последовательность во все издаваемые ими СОС.

Синтаксис последовательности «*authorityKeyIdentifier*» представлен в §4.2.1.1.

§5.2.2 Последовательность «issuerAltName»

Эта последовательность «*issuerAltName*» (альтернативное имя издателя) включает дополнительные параметры подлинности, указывающие на издателя СОС. К таким параметрам относятся адрес электронной почты («*rfc822Name*»), DNS-имя, IP-адрес и URI-идентификатор. Одновременно могут использоваться несколько частей формата наименования и несколько форматов имён. Всякий раз, когда используются такие параметры подлинности, обязательно должна использоваться последовательность «*issuerAltName*». Тем не менее, DNS-имя

может быть представлено в субполе «*Issuer*» с использованием атрибута «*domainComponent*» (§4.1.2.4).

Целесообразно, чтобы издатели СОС, придерживающиеся данного стандарта, помечали последовательность «*issuerAltName*» как «не критичную».

Описание OID и синтаксиса для данной последовательности представлено в §4.2.1.7.

§5.2.3 Последовательность «CRLNumber»

Последовательность «*CRLNumber*» (номер СОС) является не критичной и указывает на монотонно возрастающий последовательный номер, соответствующий конкретной области применения СОС и его издателю. Эта последовательность позволяет пользователям легко определить, когда конкретный СОС отменил другой СОС. Номера СОС также обеспечивают идентификацию взаимодополняющих заполняемых и усечённых СОС. Издатели СОС, придерживающиеся данного стандарта, обязаны включать эту последовательность во все СОС и помечать её как «не критичную».

Если издатель СОС формирует усечённые СОС как дополнение к заполняемым СОС в интересах конкретной области применения, то заполняемые и усечённые СОС должны, в обязательном порядке, совместно использовать одну числовую последовательность (последовательность номеров). Если усечённый и заполняемый СОС используются в одной и той же области применения и выпущены в одно и то же время, то они должны, в обязательном порядке содержать один и тот же последовательный номер и предоставлять одну и ту же информацию об отзыве сертификатов. Т.е., сочетание усечённого и соответствующего заполняемого СОС должно обеспечивать одну и ту же информацию об отзыве сертификатов, как одновременно изданные заполняемые СОС.

Если издатель СОС формирует два СОС (два заполняемых СОС, два усечённых СОС или заполняемый и усечённый СОС) для одной и той же области применения, но в разное время, то два СОС не должны иметь, в обязательном порядке, один и тот же номер СОС. Т.е., если субполе «*thisUpdate*» (§5.1.2.4) в в

обоих СОС не совпадает, то номера СОС должны быть, в обязательном порядке, разными.

Исходя из указанных выше требований, номера СОС могут достаточно длинными целыми числами. Субъекты, проверяющие номера СОС, обязаны обрабатывать значения последовательности «*CRLNumber*» длиной до 20 октетов. Издатели СОС, придерживающиеся данного стандарта, обязаны не использовать значения последовательности «*CRLNumber*» длиной более 20 октетов.

```
id-ce-cRLNumber          OBJECT IDENTIFIER ::=        { id-ce 20 }
CRLNumber ::=            INTEGER (0..MAX)
```

§5.2.4 Индикатор усечённого СОС

Последовательность «*deltaCRLIndicator*» представляет собой критичное расширение СОС, которое указывает на СОС, который выступает в роли усечённого СОС. Усечённые СОС, скорее всего, содержат обновлённые данные по отзыву сертификатов, относительно ранее распространённой информации об аннулировании, а не всю информацию, которая могла бы присутствовать в заполняемом СОС. Использование усечённых СОС в некоторых прикладных системах и ИТС может существенно снизить нагрузку на сеть и время обработки. Как правило, усечённые СОС меньше, чем СОС, которые они дополняют, и поэтому прикладные системы, получающие усечённые СОС, требуют гораздо меньшую пропускную способность сети по сравнению с прикладными системами, получающими соответствующие заполняемые СОС. Прикладные системы и ИТС, хранящие информацию об отзыве сертификатов в формате, которые не совпадает со структурой СОС, могут добавлять новую информацию об аннулировании сертификатов в локальные БД без обработки информации.

Последовательность «*deltaCRLIndicator*» содержит всего лишь одно значение типа «*BaseCRLNumber*». Номер СОС идентифицирует СОС, который заполняется для конкретной области применения, и который использовался в качестве «стартовой точки» при формировании данного усечённого СОС. Издатель СОС, придерживающийся данного стандарта, обязан публиковать спра-

вочный базовый СОС в качестве заполняемого СОС. А усечённый СОС должен содержать все новые данные относительно состояния аннулирования сертификатов для той же самой области применения. Сочетание из усечённого и справочного базового СОС эквивалентно заполняемому СОС, для прикладной сферы применения, в момент опубликования усечённого СОС.

Когда издатель СОС, придерживающийся данного стандарта, формирует усечённый СОС, тогда этот СОС должен содержать, в обязательном порядке, субполе «*deltaCRLIndicator*».

Если издан усечённый СОС, то он должен включать, в обязательном порядке, одни и те же наборы причин отзыва сертификатов и сертификатов, которые были охвачены базовым СОС, на который ссылается усечённый СОС. Т.е. область применения усечённого СОС должна быть той же самой, как и у заполняемого СОС, на который ссылается усечённый СОС, как на базовый. Справочный базовый СОС и усечённый СОС должны, в обязательном порядке, либо не включать последовательность «*issuingDistributionPoint*», либо включать идентичные последовательности «*issuingDistributionPoint*». Кроме того, издатель СОС обязан использовать один и тот же закрытый ключ для подписания усечённого СОС и любого заполняемого СОС, который может использоваться для обновления.

Прикладная система, обслуживающая усечённые СОС, может формировать СОС, который будет являться заполняемым для конкретной области применения за счёт совмещения усечённого СОС для той же области применения, либо с изданным СОС, являющимся заполняемым для этой же области применения, либо с локально сформированным СОС, также являющимся заполняемым для этой же области применения.

Когда усечённый СОС совмещён с заполняемым СОС или с локально сформированным СОС, тогда итоговый локально сформированный СОС имеет номер, который указан в последовательности «*CRLNumber*» из усечённого СОС, используемого в данной конструкции. Кроме того, итоговый локально сформированный СОС содержит субполя «*thisUpdate*» и «*nextUpdate*», время в

которых совпадает со значениями в аналогичных субполях усечённого СОС, используемого в данной конструкции. Также, локально сформированный СОС наследует последовательность «*issuingDistributionPoint*» из усечённого СОС.

Заполняемый и усечённый СОС могут быть совмещены только тогда, когда выполнены следующие четыре условия:

a) заполняемый и усечённый СОС выпущены одним и тем же издателем;

b) заполняемый и усечённый СОС имеют одну и ту же область применения. Два СОС имеют одну и ту же область применения, если выполняется одно из следующих двух условий:

1) последовательность «*issuingDistributionPoint*» отсутствует в обоих СОС (в заполняемом и усечённом);

2) последовательность «*issuingDistributionPoint*» представлена в обоих СОС (в заполняемом и усечённом), и значения в этих последовательностях совпадают;

c) номер СОС заполняемого СОС равен или больше номера «*BaseCRLNumber*», указанного в усечённом СОС. Т.е., заполняемый СОС содержит (как минимум) все данные об отозванных сертификатах, которые содержаться в справочном базовом СОС;

d) номер СОС заполняемого СОС меньше номера усечённого СОС. Т.е. усечённый СОС следует за заполняемым СОС в номерной последовательности.

Издатели СОС обязаны гарантировать, что сочетание усечённого СОС и любого другого соответствующего заполняемого СОС точно отражает текущее состояние отзыва сертификатов. Издатель СОС обязан включать в усечённый СОС (в рамках области его применения) запись о каждом сертификате, состояние которого изменилось с момента формирования справочного базового СОС, следующим образом:

a) если сертификат отозван по причине, входящей в область применения СОС, то он указывается в этом усечённом СОС как аннулированный;

b) если сертификат — действующий и был в списке справочного базового СОС или любого последующего СОС с указанием кода причины «*certificate-*

Hold», а код причины «*certificateHold*» указан в усечённом СОС, то он указывается в этом усечённом СОС как аннулированный с кодом причины отзыва «*removeFromCRL*»;

c) если сертификат аннулирован и причина отзыва выходит за пределы области применения СОС, но в то же время, сертификат был указан в справочном базовом СОС или в любом последующем СОС с кодом причины, входящей в область применения этого СОС, то он указывается в этом усечённом СОС как аннулированный без кода причины отзыва;

d) если сертификат аннулирован по причине, выходящей за пределы области применения СОС, и он не был указан, ни в справочном базовом СОС, ни в любом последующем СОС с кодом причины, входящей в область применения этого СОС, то он не включается в список этого усечённого СОС.

Состояние сертификата считается изменившимся если:

- он аннулирован (по любой причине, указанной в «*certificateHold*»);
- он изъят из СОС;
- причина его аннулирования изменилась.

Следует указывать сертификат в усечённом СОС с кодом причины отзыва «*removeFromCRL*» даже если сертификат не был указан в справочном базовом СОС. Если сертификат был включён в любой СОС, изданный после основного, но до выпуска этого усечённого СОС и затем был изъят из СОС, то он должен быть указан, в обязательном порядке, в усечённом СОС с кодом причины отзыва «*removeFromCRL*».

Издатель СОС может дополнительно включить сертификат в усечённый СОС с кодом причины отзыва «*removeFromCRL*», если время «*notAfter*», указанное в сертификате, превышает время «*thisUpdate*», указанное в усечённом СОС, и сертификат был включён в справочный базовый СОС или в любой другой СОС, изданный после выхода базового СОС, но раньше данного усечённого СОС.

Если извещение об отзыве сертификата появляется в СОС впервые, то существует вероятность того, что период действия сертификата истечёт рань-

ше, чем будет выпущен следующий заполняемый СОС для той же самой области применения. В этом случае, извещение об отзыве должно включаться, в обязательном порядке, во все последующие усечённые СОС до тех пор, пока извещение об отзыве содержится, по крайней мере, в одном однозначно изданном заполняемом СОС в интересах данной области применения.

Прикладная система, обрабатывающая усечённые СОС, обязана формировать текущий заполняемый СОС на основе совмещения ранее изданного заполняемого СОС и самого последнего текущего усечённого СОС. Кроме того, прикладная система, обрабатывающая усечённые СОС, может сформировать текущий заполняемый СОС на основе совмещения ранее локально сформированного заполняемого СОС и текущего усечённого СОС. Усечённый СОС считается текущим СОС, если текущее время укладывается в интервал между временны̀ми метками, содержащимися в субполях «*thisUpdate*» и «*nextUpdate*». В некоторых ситуациях издатель СОС может выпустить один или несколько усечённых СОС ещё до наступления момента времени, указанного в субполе «*nextUpdate*». Если прикладная система «столкнётся» с несколькими текущими усечёнными СОС, то ей целесообразно принять во внимание СОС, в котором указано самое позднее значение в субполе «*thisUpdate*», т.е. этот СОС становится самым последним текущим усечённым перечнем.

id-ce-deltaCRLIndicator	OBJECT IDENTIFIER ::=	{ id-ce 27 }
BaseCRLNumber ::=	CRLNumber	

§5.2.5Точка распространения СОС

Последовательность «*issuingDistributionPoint*» представляет собой критичное расширение СОС, которое указывает на точку распространения СОС и область применения соответствующего СОС, а также устанавливает, включает ли СОС только отозванные сертификаты конечных пользователей, только атрибутные сертификаты или только сертификаты, аннулированные в соответствие с ограниченной группой кодов причин отзыва. Несмотря на то, что последовательность «*issuingDistributionPoint*» является критичной, от прикладных систем

и ИТС, придерживающихся данного стандарта, не требуется обрабатывать данный информационный элемент. Тем не менее, прикладные системы и ИТС, которые не обрабатывают данную последовательность, обязаны, либо проверять состояние любого сертификата, не представленного в данном СОС из-за того, что он не известен, либо использовать другой СОС, который не включает каких-либо неизвестных критичных расширений.

Издатель СОС подписывает выпущенный им СОС с использованием своего закрытого ключа. В точках распространения СОС отсутствуют свои собственные пары криптоключей. Если СОС храниться в репозитарии (сервере) СЕК-службы (Рекомендация ITU-T X.500), то он храниться в элементе каталога, соответствующем точке распространения СОС, которая может отличаться от элемента каталога, соответствующего издателю СОС.

Кода причин отзыва, связанные с точкой распространения, должны указываться, в обязательном порядке, в субпоследовательности «*onlySomeReasons*». Если субпоследовательность «*onlySomeReasons*» не представлена, то субпоследовательность «*distributionPoint*» должна включать, в обязательном порядке, отзывы сертификатов со всеми кодами причин. УЦ могут воспользоваться точками распространения СОС для разделения СОС, либо для скомпрометированных сертификатов, либо для отзыва сертификатов обычным порядком. В таком случае, отзывы сертификатов с кодом причины «*keyCompromise*» (*1*), «*cACompromise*» (*2*) и «*aACompromise*» (*8*) содержатся в одной точке распространения СОС, а отзывы сертификатов другими кодами причин содержатся в другой точке распространения СОС.

Если СОС включает последовательность «*issuingDistributionPoint*» с субпоследовательностью «*onlySomeReasons*», то для каждого аннулированного сертификата, входящего в область применения СОС, должна быть указана причина его отзыва, которая не относится категории неизвестных (*unspecified*). Указание причины отзыва используется для определения в каком СОС представлен аннулированный сертификат, тем не менее, включение расширения записи СОС «*reasonCode*» в соответствующую запись СОС не обязательно.

Синтаксис и семантика субпоследовательности «*distributionPoint*» те же самые, как и в последовательности «*distributionPoint*» субполя «*cRLDistributionPoints*» (§4.2.1.13). Если субпоследовательность «*distributionPoint*» представлена, то она должна включать, в обязательном порядке, по крайней мере, одно из наименований, содержащихся в последовательности «*distributionPoint*» субполя «*cRLDistributionPoints*» каждого сертификата, который входит в сферу применения данного СОС. В субпоследовательности «*distributionPoint*» СОС и в последовательности «*distributionPoint*» сертификата должен использоваться, в обязательном порядке, один и тот же тип кодирования.

Если субпоследовательность «*distributionPoint*» отсутствует, то СОС должен содержать, в обязательном порядке, записи для всех аннулированных непросроченных сертификатов, выпущенных издателем СОС. В противном случае, СОС должен содержать, в обязательном порядке, записи для всех аннулированных непросроченных сертификатов, которые входят в сферу применения СОС.

Если область применения СОС включает только сертификаты, выпущенные издателем СОС, то субпоследовательность «*indirectCRL*» (содержит логический тип данных, «*boolean*») должна содержать, в обязательном порядке, значение «*FALSE*» (не верно). В противном случае, если область применения СОС включает сертификаты, выпущенные одним или несколькими УЦ, но отличными от издателя СОС, то субпоследовательность «*indirectCRL*» (содержит логический тип данных, «*boolean*») должна содержать, в обязательном порядке, значение «*TRUE*» (верно). УЦ, ответственный за каждую запись, указывается с помощью расширения записи СОС «*certificateIssuer*» (§5.3.3).

Если область применения СОС включает только СЕРТ|ОК конечных пользователей, то субпоследовательность «*onlyContainsUserCerts*» должна содержать, в обязательном порядке, значение «*TRUE*» (верно). Если же область применения СОС включает только сертификаты УЦ, то субпоследовательность «*onlyContainsCACerts*» должна содержать, в обязательном порядке, значение «*TRUE*» (верно). Если одна из субпоследовательностей, либо «*onlyContainsUs-*

erCerts», либо «*onlyContainsCACerts*» содержит значение «*TRUE*» (верно), то область применения СОС должна, в обязательном порядке, не включать сертификаты 1-ой и 2-ой версий. Издатели СОС, придерживающиеся данного стандарта, обязаны устанавливать в субпоследовательность «*onlyContainsAttributeCerts*» (содержит логический тип данных, «*boolean*») значение «*FALSE*» (не верно).

Издатели СОС, придерживающиеся данного стандарта, обязаны не выпускать сертификаты, в которых последовательность «*issuingDistributionPoint*», имеющая DER-кодировку, содержит пустое значение. Т.е., если все субпоследовательности «*onlyContainsUserCerts*», «*onlyContainsCACerts*», «*indirectCRL*» и «*onlyContainsAttributeCerts*» содержат значение «*FALSE*» (не верно), то одна из субпоследовательностей, либо «*distributionPoint*», либо «*onlySomeReasons*» должна быть представлена, в обязательном порядке.

```
id-ce-issuingDistributionPoint      OBJECT IDENTIFIER ::= { id-ce 28 }

IssuingDistributionPoint ::=        SEQUENCE {
        distributionPoint               [0]     DistributionPointName OPTIONAL,
        onlyContainsUserCerts           [1]     BOOLEAN DEFAULT FALSE,
        onlyContainsCACerts             [2]     BOOLEAN DEFAULT FALSE,
        onlySomeReasons                 [3]     ReasonFlags OPTIONAL,
        indirectCRL                     [4]     BOOLEAN DEFAULT FALSE,
        onlyContainsAttributeCerts      [5]     BOOLEAN DEFAULT FALSE }

        -- не более, чем одна из субпоследовательностей «onlyContainsUserCerts»,
        -- «onlyContainsCACerts» и «onlyContainsAttributeCerts» может содержать
        -- значение «TRUE».
```

§5.2.6 «Свежайший» СОС (так называемая точка распространения СОС)

Последовательность «*freshestCRL*» определяет как получить информацию об усечённом СОС для данного заполняемого СОС. Издатели СОС, придерживающиеся данного стандарта, обязаны помечать эту последовательность как «не критичную». Эта последовательность не должна, в обязательном порядке, присутствовать в усечённых СОС.

Синтаксис, используемый в данной последовательности, такой же, как и в субполе «*cRLDistributionPoints*» поля «Расширения» сертификата (§4.2.1.13). Тем не менее, в данном контексте только субпоследовательность «*distribution-*

Point» имеет важное значение. Субпоследовательности «*reasons*» и « *cRLIssuer*» должны быть исключены, в обязательном порядке, из последовательности «*freshestCRL*».

Каждое наименование точки распространения определяет место в сети, в котором может быть найден усечённый СОС для данного заполняемого СОС. Область применения таких усечённых СОС должна быть, в обязательном порядке, такой же, как и у данного заполняемого СОС. Содержание последовательности «*freshestCRL*» используется только для определения местоположения усечённых СОС. Оно не используется для подтверждения подлинности СОС или усечённых СОС, на которые имеются ссылки. Для данной последовательности «*freshestCRL*» используются правила кодирования, которые установлены для последовательностей «*distributionPoint*» в субполях «*cRLDistributionPoints*» (§4.2.1.13).

```
id-ce-freshestCRL       OBJECT IDENTIFIER ::= { id-ce 46 }
FreshestCRL ::=         CRLDistributionPoints
```

§5.2.7 Доступ к информации УЦ

Далее рассматривается применение последовательности «*authorityInformationAccess*» в СОС. Синтаксис и семантика, используемые в данной последовательности, аналогичны тем, которые используются в субполе «*crlExtensions*» СОС (§4.2.2.1).

Эта последовательность должна, в обязательном порядке, маркироваться как «не критичная».

Когда последовательность «*authorityInformationAccess*» представлена в СОС, тогда он должна, в обязательном порядке, включать, по крайней мере, одно описание доступа (субпоследовательность «*AccessDescription*»), содержащее идентификатор «*id-ad-caIssuers*» в качестве способа доступа «*accessMethod*». Идентификатор «*id-ad-caIssuers*» используется тогда, когда доступная информация содержит список сертификатов, которые могут использоваться при проверке подписи СОС (т.е., сертификаты, которые содержат имя владельца сер-

тификата, сравниваемое с именем издателя, указанного в СОС, и которые содержат открытый ключ владельца сертификата, соответствующий закрытому ключу, используемому при подписании СОС). Типы способов доступа, отличающиеся от тех, которые определяют идентификаторы «*id-ad-caIssuers*», должны быть исключены, в обязательном порядке. По крайней мере, одна запись в субпоследовательности «*AccessDescription*» должна определять точку доступа «*accessLocation*», для которой используется URI-идентификатор совместно HTTP- или LDAP-протоколом (RFC-2616, RFC-4516).

Когда информация предоставляется на основе HTTP- или FTP-протокола, тогда запись «*accessLocation*» должна, в обязательном порядке, включать идентификатор «*uniformResourceIdentifier*», а URI-идентификатор должен, в обязательном порядке, указывать, либо на одиночный сертификата в DER-кодировке (RFC-2585), либо на совокупность сертификатов в BER- и DER-кодировке, доставляемых с помощью CMS-сообщений типа «*certs-only*» (RFC-2797).

Прикладные системы, придерживающиеся данного стандарта и использующие HTTP- или FTP-протокол для обеспечения доступа к сертификатам, обязаны предоставлять доступ к индивидуальным сертификатам, закодированным по DER-правилам, а также целесообразно, чтобы такие прикладные системы обеспечивали обмен CMS-сообщениями (тип «*certs-only*»).

Целесообразно, чтобы прикладные системы, функционирующие с использованием URI-идентификаторов при обеспечении доступа к HTTP-серверу, указывали тип доставки «*application/pkix-cert*» (RFC-2585) в поле заголовка «*content-type*» ответного сообщения на запрос одиночного сертификата, закодированного по DER-правилам. Кроме того, целесообразно, чтобы такие прикладные системы указывали тип доставки «*application/pkcs7-mime*» (RFC-2797) в поле заголовка «*content-type*» ответного сообщения при обмене CMS-сообщениями («*certs-only*»). При использовании FTP-протокола, целесообразно, чтобы имя файла, содержащего одиночный сертификат, закодированный по DER-правилам, включало суффикс «*.cer*» (RFC-2585), а имя файла, содержащего CMS-сообщение («*certs-only*»), — «*.p7c*» (RFC-2797). Касаясь клиентов, то

они могут использовать расширение «тип доставки» («*media type*») или «файл» («*file*») как указатель на содержание, но, в то же время, не целесообразно, чтобы оно зависело только от наличия корректного расширения «тип доставки» или «файл» в ответе сервера.

Если запись «*accessLocation*» является наименованием в формате «*directoryName*», то прикладная система должна получать необходимую информацию из любого СЕК-сервера, который настраивается, исходя из локальных условий. Когда для подтверждения подлинности сертификатов и СОС используется открытый ключ одного УЦ, тогда необходимый сертификат УЦ размещается в атрибуте «*crossCertificatePair*» и/или «*cACertificate*», как это определено в стандарте RFC-4523. Когда для подтверждения подлинности сертификатов и СОС используется различные открытые ключи, тогда необходимый сертификат УЦ размещается в атрибуте «*userCertificate*», как это определено в стандарте RFC-4523. Таким образом, прикладные системы, обрабатывающие формат имени «*directoryName*», включённого в запись «*accessLocation*», обязаны находить необходимый сертификат в одном из этих трёх атрибутов. Протокол, используемый прикладной системой (прикладным процессом) для доступа к СЕК-сегменту (например, DAP- или LDAP-протокол), выбирается исходя из локальных условий.

Когда доступ к информации обеспечивается с помощью LDAP-протокола, тогда целесообразно, чтобы запись «*accessLocation*» представляла собой URI-идентификатор («*uniformResourceIdentifier*»). URI-идентификатор LDAP-сервера (RFC-4516) должен, в обязательном порядке, включать поле «<*dn*>», содержащее уникальное имя записи, содержащей сертификатов, он должен, в обязательном порядке, включать субпоследовательность «<*attributes*>», содержащая список описаний атрибутов, которые содержат сертификаты или пары кросс-сертификатов в DER-коде (RFC-4523), и целесообразно, чтобы он включал поле «<*host*>» (например, <*ldap://ldap.example.com/cn=CA,dc=example,dc=com?cACertificate;binary,crossCertificatePair;binary*>). Пропуск поля «<*host*>» (например, <*ldap://cn=exampleCA,dc=example,dc=com?cACertificate;binary*>) может

быть эффективным тогда, когда клиент обладает какими-нибудь априорными знаниями для соединения с соответствующим сервером.

§5.3 Расширения записей COC

Расширения записей COC определены в стандартах ISO/IEC, ITU-T и ANSI X9 для COC, формат и конструкция которого установлены Рекомендацией ITU-T X.509 версия 2. Эти расширения устанавливают способы для связи дополнительных атрибутов с записями COC (X.509 и X9.55). Формат и семантическая конструкция COC, которые установлены Рекомендацией ITU-T X.509 версия 2, также позволяют PKI-сообществам вводить частные расширения записей COC для доставки информации, являющейся уникальной для таких сообществ. Каждое расширение в записи COC может быть помечено как «критичное» или «не критичное». Если COC содержит критичное расширение записи, которое не может быть обработано прикладным процессом, то прикладная система обязана не использовать этот COC при определении состояния каких-либо сертификатов. Тем не менее, прикладные системы и ИТС могут игнорировать не распознаваемые не критичные расширения записей COC.

Далее представлены рекомендуемые расширения, используемые в записях COC в рамках Интернет/PKI-инфраструктуры, и стандартные точки размещения служебной PKI-информации. PKI-сообщества могут использовать дополнительные расширения записей COC. Тем не менее, следует внимательно подходить к вопросу внедрения любого критического расширения записи в COC, которое может быть использовано в общем контексте.

Обработка расширений записей COC, представленная ниже, является не обязательной (дополнительной) для издателей COC и прикладных систем, придерживающихся положениям данного стандарта. Однако, целесообразно, чтобы издатели COC включали коды причин (§5.3.1) и даты истечения сроков действия сертификатов (§5.3.2) каждый раз, когда такая информация доступна.

§5.3.1 Код причины

Код причины «*reasonCode*» является не критичным расширением записи СОС, которое определяет причину отзыва сертификата. От издателей СОС строго требуется включать коды значимых причин в записи СОС. Однако, целесообразно, чтобы при отсутствии причины отзыва сертификата расширение записи СОС «*reasonCode*» содержало значение «*unspecified*» (0), т.е. не определено.

Значение «*removeFromCRL*» (8) в расширении записи СОС «*reasonCode*» может быть установлено только в усечённых СОС, что означает удаление сертификата из СОС вследствие, либо просроченности сертификата, либо удаления привязки к своему владельцу. Все другие коды причин могут использоваться в любом СОС, указывая на то, что определённый сертификат должен рассматриваться как аннулированный.

```
id-ce-cRLReasons          OBJECT IDENTIFIER ::= { id-ce 21 }

        -- reasonCode ::= { CRLReason }

CRLReason ::= ENUMERATED {
        Unspecified                 (0),
        keyCompromise               (1),
        cACompromise                (2),
        affiliationChanged          (3),
        superseded                  (4),
        cessationOfOperation        (5),
        certificateHold             (6),
                -- значение «7» не используется
        removeFromCRL               (8),
        privilegeWithdrawn          (9),
        aACompromise                (10) }
```

§5.3.2 Дата окончания срока действия сертификата

Дата окончания срока действия сертификата «*invalidityDate*» является не критичным расширением записи СОС, указывающее на дату, которая известна или предполагается, когда закрытый ключ был скомпрометирован или, в противном случае, когда этот сертификат стал не действительным. Эта дата может иметь более раннее значение по сравнению с датой отзыва, содержащейся в записи СОС. Дата отзыва представляет собой время, когда УЦ аннулировал сертификат (провёл процедуру отзыва сертификата). Когда дата отзыва вначале помещается в СОС самим издателем СОС, тогда дата окончания срока действия

сертификата может предшествовать дате выпуска более ранних СОС, но всё-таки не целесообразно, чтобы дата отзыва предшествовала дате выпуска более ранних СОС. Всякий раз, когда эта информация становится доступной, строго рекомендуется, чтобы издатели СОС предоставили её пользователям СОС.

Эта последовательность «*invalidityDate*» должна содержать значение в кодировке «*GeneralizedTime*», а это значение обязательно должно быть представлено как время по Гринвичу (*Greenwich Mean Time*). Правила кодирования и формат этого значения времени совпадают с правилами и форматом, представленными в §4.1.2.5.2.

```
id-ce-invalidityDate        OBJECT IDENTIFIER ::= { id-ce 24 }
InvalidityDate ::=          GeneralizedTime
```

§5.3.3 Издатель сертификата

Это расширение записи СОС «*certificateIssuer*» указывает на издателя сертификата, который связан (фактически является автором) с записью в косвенном СОС. Т.е. СОС, который содержит индикатор косвенного СОС, установленный в последовательности «*issuingDistributionPoint*». Если последовательность «*certificateIssuer*» представлена, то она содержит одно или несколько имён из поля «*Issuer*» и/или субполя «*issuerAltName*» сертификата, который соответствует записи СОС. Если же последовательность «*certificateIssuer*» не представлена в начальной записи косвенного СОС, то издатель сертификата, по умолчанию, является издателем СОС. Если в последующих записях косвенного СОС эта последовательность не представлена, то издатель сертификата, соответствующий конкретной записи, является тем же самым, по отношению к предшествующей записи. Эта последовательность имеет следующий формат:

```
id-ce-certificateIssuer     OBJECT IDENTIFIER ::= { id-ce 29 }
CertificateIssuer ::=       GeneralNames
```

Издатели СОС, придерживающиеся данного стандарта, обязаны включать в эту последовательность уникальное имя (*distinguished name*) из поля «*Issuer*»

сертификата, который соответствует этой записи СОС. Кодирование уникального имени должно быть, в обязательном порядке, идентичным кодированию, используемому в сертификате.

Издатели СОС обязаны помечать последовательность «*certificateIssuer*» как критичную, так как прикладная система или ИТС, игнорирующая данное расширение записи СОС, не могли бы корректно привязывать записи СОС к сертификатам. Данный стандарт рекомендует, что бы прикладные системы и ИТС распознавали данное расширение записей СОС.

VI ПОДТВЕРЖДЕНИЕ ПОДЛИННОСТИ МАРШРУТА СЕРТИФИКАЦИИ

Процедуры подтверждения подлинности (ПрПП) МС в рамках Интернет/PKI-инфраструктуры основаны на алгоритме, представленном в Рекомендации ITU-T X.509 (X.509-алгоритм). Обработка МС подразумевает проверку привязки уникального имени владельца сертификата и/или его альтернативного имени к его открытому ключу. Такая связка ограничивается некоторыми условиями, содержащимися в сертификате, который включает маршрут и входные данные, устанавливаемые проверяющей стороной. Субполя «*basicConstraints*» и «*policyConstraints*» обеспечивают логическую обработку МС, что позволяет автоматизировать процесс принятия решения.

Далее рассматривается X.509-алгоритм ПрПП МС. От прикладных систем и ИТС, придерживающихся данного стандарта, не требуется применять стандартный X.509-алгоритм, но они обязаны использовать эквивалентную ПрПП маршрутов, обеспечивающую функциональную совместимость с внешней процедурной характеристикой, реализуемой X.509-алгоритмом, и приводящую к результату, который аналогичен результату при выполнении X.509-алгоритма. Соответствующая прикладная система или ИТС может использовать любой алгоритм сколь угодно долго, но при условии, что он будет приводить к корректному результату.

Ниже описывается основная ПрПП маршрута. Приемлемые маршруты начинаются с сертификатов, изданных доверенным «замыкающим УЦ» («*trust anchor*[°]»). Для алгоритма необходимы открытый ключ УЦ, наименование УЦ и ограничения на установку маршрутов, подлинность которых может быть подтверждена с использованием этого криптоключа.

Выбор доверенного замыкающего УЦ (ДЗУЦ) — это дело политики сертификации. ДЗУЦ не может быть корневой УЦ в иерархической PKI-инфраструктуре, УЦ, который издал собственный(е) сертификат(ты) проверяющего, или любой другой УЦ в PKI-инфраструктуре сетевого типа. ПрПП маршрута неизменна и не зависит от выбора ДЗУЦ. Кроме того, различные прикладные системы могут доверять различным ДЗУЦ или могут признавать маршруты, которые начинаться с любой группы ДЗУЦ.

Далее будут представлены способы использования алгоритма ПрПП маршрута в специфических прикладных системах и ИТС, а также итерации, необходимые для определения, является ли сертификат отозванным, когда ведение СОС является способом аннулирования, используемым издателем сертификатов.

§6.1 Основная процедура подтверждения подлинности маршрута

Далее рассматривается алгоритм обработки маршрута, представленный в Рекомендации ITU-T X.509. Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны включать процедуру обработки маршрутов в соответствие с Рекомендацией ITU-T X.509, которая является функционально совместима (эквивалентна) с внешней процедурной характеристикой, реализуемой X.509-алгоритмом. Тем не менее, обработка некоторых расширений сертификата, реализуемая X.509-алгоритмом, является не обязательной для совместимых прикладных систем и ИТС. Клиенты Интернет/PKI-инфраструктуры, которые

[°] Буквальный перевод «доверенный якорь». Но в связи с тем, что маршрут сертификации представляет собой цепочку УЦ (изданных ими сертификатов), то под «якорем» понимается замыкающий цепочку УЦ (изданный им сертификат). По одной из версий перевода, под термином «*anchor*» понимается замыкающий участник в каждой из команд, соревнующихся в перетягивании каната. Замыкающим разрешают сделать углубления в земле для устойчивости (т.е. своеобразный «якорь»).

не обрабатывают такие расширения, могут исключить соответствующие итерации из X.509-алгоритма ПрПП маршрута.

Например клиенты, от которых не требуется обработка субполя «*policyMappings*» (отображения политики). Такие клиенты могут пропускать итерации ПрПП маршрута, в которых обрабатываются отображения политики. Следует заметить, что клиенты Интернет/PKI-инфраструктуры обязаны удалять сертификат, если он содержит необслуживаемое ими расширение, помеченное как критичное.

Несмотря на то, что описания сертификата и СОС, представленные в разделах IV и V данного стандарта соответственно, устанавливают логическую конструкцию (формат и кодирование) полей и расширений сертификатов и СОС, которые предназначены для соответствующего применения в Интернет/PKI-инфраструктуре, рассматриваемый далее X.509-алгоритм не ограничивает получение сертификатов и СОС, которые соответствуют логическим конструкциям данного стандарта. Более того, X.509-алгоритм включает только проверки того, что МС является приемлемым в соответствие с Рекомендацией ITU-T X.509, а также не включает проверок того, что сертификаты и СОС удовлетворяют требованиям данного стандарта. Несмотря на то, что X.509-алгоритм мог быть расширен и включал бы проверки соответствия сертификатов и СОС описаниям, представленным в разделах IV и V данного стандарта, последний настоятельно рекомендует такие проверки не включать.

X.509-алгоритм подтверждает подлинность сертификата, что касается текущих даты и времени. Прикладные системы и ИТС, придерживающиеся данного стандарта, также могут проводить ПрПП по отношению к некоторой точке в маршруте. (Примечание. В данном случае способы подтверждения подлинности сертификата относительно времени, которое вышло за пределы периода действия сертификата, не приемлемы.)

ДЗУЦ (ДЗУЦ-сертификат) рассматривается как входные данные X.509-алгоритма. Не существует требований, чтобы один и тот же ДЗУЦ-сертификат

использовался в ПрПП всех МС. Различные ДЗУЦ-сертификаты могут использоваться в ПрПП различных маршрутов.

Основная цель ПрПП заключается в проверке привязки (связки) уникального или альтернативного имени владельца сертификата к его же открытому ключу, представленном в целевом сертификате, на основе открытого ключа ДЗУЦ. В большинстве случаев, целевой сертификат будет представлять собой сертификат конечного пользователя, но целевой сертификат может быть и сертификатом УЦ, пока открытый ключ владельца сертификата предназначен для решения любых других задач, отличных от проверки подписи в СЕРТ|ОК. Проверка связки между именем и открытым ключом владельца сертификата требует получения последовательности сертификатов, которые обеспечивают такую связку. Процедура получения такой последовательности сертификатов в данном стандарте не рассматривается.

Для достижения этой цели, ПрПП маршрутов проверяет, среди прочего, что предполагаемый МС (последовательность из n сертификатов) удовлетворяет следующим условиям:

a) для всех x в {1, ..., n-1}, владелец сертификата x является издателем сертификата x+1;

b) сертификат 1 издан ДЗУЦ;

c) сертификат n является сертификатом, подлинность которого должна быть подтверждена (т.е. целевой сертификат);

d) для всех x в {1, ..., n}, сертификат был действителен на момент времени, о котором идёт речь.

Сертификат должен присутствовать в предполагаемом МС не более одного раза, в обязательном порядке.

Когда ДЗУЦ-сертификат представлен в форме само-подписанного сертификата, тогда этот само-подписанный сертификат не включается в предполагаемый МС как его «звено». Информация о ДЗУЦ предоставляется в качестве входных данных алгоритма ПрПП МС.

Соответствующая часть маршрута, тем не менее, может быть не приемлемой для всех прикладных систем. Более того, прикладная система может расширить алгоритм до нужного предела, определяемого совокупностью приемлемых маршрутов. ПрПП маршрута также устанавливает совокупность политик, которые допустимы для этого МС, основываясь при этом на данных в субполях «*certificatePolicies*», «*policyMappings*», «*policyConstraints*» и «*inhibitAnyPolicy*». Для выполнения этого алгоритм ПрПП маршрута строит дерево приемлемых политик. Если совокупность политик сертификации, которые являются допустимыми для этого маршрута, представлена, то в результате будет сформировано дерево приемлемых политик глубиной *n*, в противном случае, будет сформировано нулевое (пустое) дерево приемлемых политик.

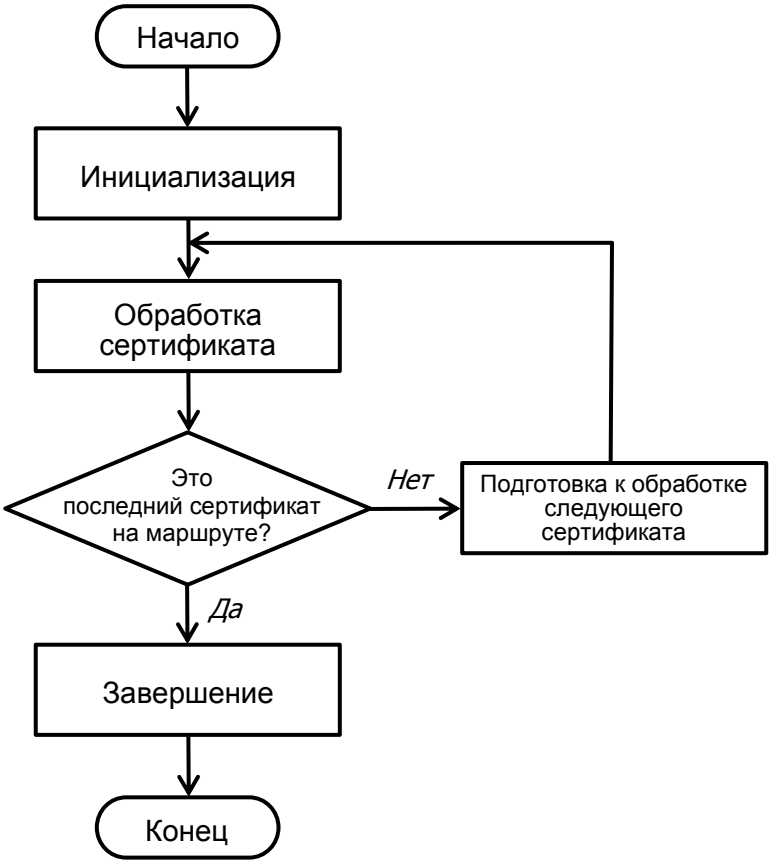

Рис. 2. Блок-схема процедуры обработки МС

Сертификат является само-изданным, если одно и тоже уникальное имя представлено в полях «*Subject*» и «*Issuer*» (два уникальных имени являются одинаковыми, если они совпадают при их сравнении по правилам, представ-

ленным в §7.1). В общем, поля «*Subject*» и «*Issuer*» в сертификатах, составивших маршрут, отличаются в каждом сертификате. Однако, УЦ может издать сертификат для самого себя с целью обеспечения смены ключей и изменений политик сертификации. Такие само-изданные сертификаты не учитываются, когда анализируются глубина маршрута или ограничения на форматы имён (субполе «*nameConstraints*»).

X.509-алгоритм включает четыре итерации (фазы):

1) инициализация;

2) основная обработка сертификата;

3) подготовка к обработке следующего сертификата;

4) завершение.

Итерации (1) и (4) выполняются только один раз, в начале и при завершении X.509-алгоритма. Итерация (2) выполняется при обработке всех сертификатов маршрута. Итерация (3) выполняется при обработке всех сертификатов маршрута, за исключение последнего сертификата. На рис. 2 представлена блок-схема X.509-алгоритма.

§6.1.1 Входные данные

Данный алгоритм предполагает, что для программы обработки МС предоставляются следующие девять входных параметров:

a) предполагаемый МС глубиной *n*;

b) текущие дата и время;

c) совокупность исходных политик пользователя («*user-initial-policy-set*»). Это совокупность идентификаторов политик сертификации, которые указывают на политики, приемлемые для пользователя сертификата. Данная совокупность содержит специальное значение в субполе «*anyPolicy*», если пользователь не имеет отношения к политике сертификации;

d) информация о ДЗУЦ («*trust anchor information*»), описывающая УЦ, который выступает в роли ДЗУЦ в МС. Такая информация включает:

1) имя доверенного издателя;

2) доверенный алгоритм для открытого ключа;

3) доверенный открытый ключ;

4) и дополнительно, параметры доверенного открытого ключа, связанного с открытым ключом.

Информация о ДЗУЦ для процедуры обработки маршрута может предоставляться в форме само-подписанного сертификата. Когда информация о ДЗУЦ предоставляется в форме сертификата, тогда имя в поле «*Subject*» используется в качестве доверенного имени издателя, а содержание поля «*subjectPublicKeyInfo*» используется как источник доверенных алгоритма для открытого ключа и самого открытого ключа. Информация о ДЗУЦ является надёжной потому, что она доставлялась для её использования в процедуре обработки МС с помощью некоторой надёжной автономной процедуры (*out-of-band procedure*). Если доверенный алгоритм для открытого ключа требует дополнительные параметры, то они предоставляются вместе с доверенным открытым ключом.

e) исходный запрет на отображение политик («*initial-policy-mapping-inhibit*»), который указывает, разрешено ли отображение политик в МС;

f) точно определённая исходная политика («*initial-explicit-policy*»), которая указывает на то, что маршрут должен быть, в обязательном порядке, приемлем хотя бы для одной из политик сертификации, представленных в совокупность исходных политик пользователя;

g) исходный запрет на использование субполя «*anyPolicy*» («*initial-any-policy-inhibit*»), который указывает, целесообразно ли обрабатывать OID в субполе «*anyPolicy*», если этот идентификатор представлен в сертификате;

h) разрешённые исходные субдеревья («*initial-permitted-subtrees*»), которые определяют для каждого формата имён (например, уникальные имена в соответствие с Рекомендацией ITU-T X.500, адреса электронной почтовой службы или IP-адреса) совокупность субдеревьев, в рамках которых каждый сертификат маршрута сертификации содержит все имена своих владельцев, должны отбрасываться, в обязательном порядке. Входные данные о разрешённых ис-

ходных субдеревьях включают несколько групп, каждая для конкретного формата (типа) наименования. Для каждого формата имени такая группа может состоять из одиночного субдерева, которое включает все имена такого формата, или одного или нескольких субдеревьев, каждое из которых определяет подгруппу имён данного формата, или такая группа может быть пустой. Если группа входных данных для некоторого формата имён является пустой, то МС будет считаться не приемлемым в том случае, когда любой сертификат в МС содержит имя такого формата;

i) недопустимые исходные субдеревья («*initial-excluded-subtrees*»), которые определяют для каждого формата имён (например, уникальные имена в соответствие с Рекомендацией ITU-T X.500, адреса электронной почтовой службы или IP-адреса) совокупность субдеревьев, в рамках которых любой сертификат маршрута сертификации не содержит имя своего владельца, могут отбрасываться. Для каждого формата имени такая группа может быть пустой или может состоять из одного или нескольких субдеревьев, каждое из которых определяет подгруппу имён данного формата. Если группа входных данных для некоторого формата имён является пустой, то имена, несоответствующие данному формату имён, исключаются.

От прикладных систем и ИТС, придерживающихся данного стандарта, не требуется группировать все такие допустимые входные данные. Например, прикладные системы и ИТС, придерживающихся данного стандарта, могут обрабатывать все МС, используя значение «*FALSE*» для входного параметра «*initial-any-policy-inhibit*» (исходный запрет на использование субполя «*anyPolicy*»).

§6.1.2 Фаза инициализации

В этой фазе формируются одиннадцать следующих переменных состояния на основе девяти входных параметров:

a) переменная «*valid_policy_tree*» (дерево допустимых политик) — это дерево политик сертификации вместе с их дополнительными определителями.

Каждое из листьев дерева представляет собой приемлемую политику в этой фазе ПрПП маршрута сертификации. Если в этой фазе ПрПП маршрута сертификации существуют приемлемые политики, то глубина дерева равна числу сертификатов в их последовательности, которая подвергается обработке. Если в этой фазе ПрПП маршрута сертификации не существуют приемлемые политики, то дерево устанавливается в ноль («*NULL*»). Как только дерево устанавливается в ноль, обработка политики прекращается.

Каждый узел в дереве допустимых политик («*valid_policy_tree*») включает три информационных объекта: приемлемую политику, набор связанных с политикой определителей и набор из одной или нескольких параметров ожидаемой политики. Если узел имеет глубину *x*, то компоненты узла имеют следующую семантику:

1) компонент «*valid_policy*» (допустимая политика) представляет собой одиночный OID политики, указывающий на приемлемую политику для маршрута длиной *x*;

2) компонент «*qualifier_set*» (набор определителей) представляет собой набор определителей политики, связанных с приемлемой политикой, указанной в сертификате *x*;

3) компонент «*expected_policy_set*» (совокупность ожидаемых политик) содержит один или несколько OID политики, которые могли бы удовлетворять этой политике, указанной в сертификате *x*+1.

Исходное значение входного параметра «*valid_policy_tree*» представляет собой одиночный узел с компонентом «*valid_policy*» (допустимая политика), имеющим значение «*anyPolicy*», пустым компонентом «*qualifier_set*» и компонентом «*expected_policy_set*», имеющим одно значение «*anyPolicy*». Считается, что этот узел имеет глубину ноль («*zero*»).

На рис. 3 представлено графическое изображение исходного состояния дерева «*valid_policy_tree*». В дальнейшем в течение обработки маршрута будет использоваться именно этот формат с целью определения изменений в дереве «*valid_policy_tree*».

b) переменная «*permitted_subtrees*» (разрешённые субдеревья) — совокупность корневых имён для каждого формата (типа) наименований (например, уникальные имена в соответствие с Рекомендацией ITU-T X.500, адреса электронной почтовой службы или IP-адреса) определяет группу субдеревьев, в рамках которых каждый сертификат маршрута сертификации содержит все имена владельцев, которая должна отбрасываться, в обязательном порядке. Эта переменная включает совокупность имён для каждого формата, а исходным значением является входной параметр «*initial-permitted-subtrees*»;

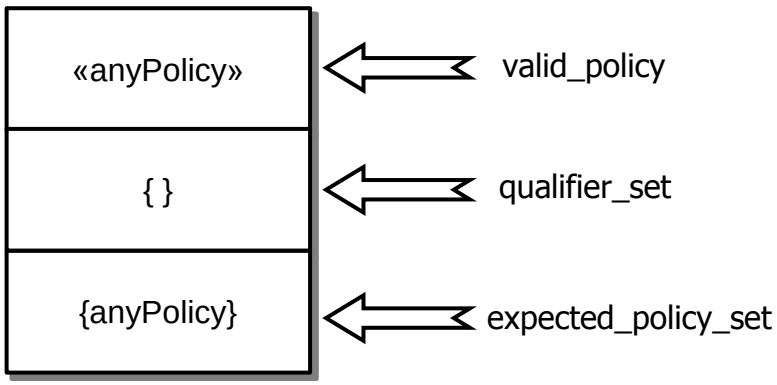

Рис. 3. Начальное значение переменной состояния дерева «*valid_policy_tree*»

c) переменная «*excluded_subtrees*» (запрещённые субдеревья) — совокупность корневых имён для каждого формата (типа) наименований (например, уникальные имена в соответствие с Рекомендацией ITU-T X.500, адреса электронной почтовой службы или IP-адреса) определяет группу субдеревьев, в рамках которых последующие сертификаты маршрута сертификации не содержат имени владельца, которая может отбрасываться. Эта переменная включает совокупность имён для каждого формата, а исходным значением является входной параметр «*initial-excluded-subtrees*»;

d) переменная «*explicit_policy*» (явная политика) — целое число, которое указывает, требуется ли не нулевое дерево «*valid_policy_tree*». Целое число определяет количество не само-изданных сертификатов, которые должны быть обработана, ещё до того, как это требование будет «выставлено». Когда требование установлено, данная переменная может быть уменьшена, но не может

быть увеличена. Т.е., если сертификат маршрута требует не нулевого дерева «*valid_policy_tree*», то в последующем сертификат не может «снять» это требование. Если параметр «*initial-explicit-policy*» установлен, то начальное значение равно 0, в противном случае, начальное значение равно *n*+1;

e) переменная «*inhibit_anyPolicy*» (запрет любой политики) — целое число, которое указывает, совпал ли идентификатор политики «*anyPolicy*» при сравнении. Это целое число определяет количество не само-изданных сертификатов, которые должны быть обработаны, прежде чем будет обработан OID в субполе «*anyPolicy*», если оно заявлено в сертификате, отличающемся от промежуточного само-подписанного сертификата, то оно игнорируется. Когда эта переменная установлена, тогда она может снижена, но не может быть увеличена. Т.е., если сертификат маршрута запрещает обработку субполя «*anyPolicy*», то последующий сертификат маршрута не может её разрешить. Если параметр «*initial-any-policy-inhibit*» установлен, то начальное значение равно 0, в противном случае, начальное значение равно *n*+1;

f) переменная «*policy_mapping*» (отображение политики) — целое число, которое указывает, разрешено ли отображение политик. Это целое число определяет количество не само-изданных сертификатов, которые должны быть обработаны, прежде чем отображение политик будет запрещено. Когда эта переменная установлена, тогда она может снижена, но не может быть увеличена. Т.е., если сертификат маршрута устанавливает, что отображение политик запрещено, то этот запрет не может быть аннулирован последующим сертификатом. Если параметр «*initial-policy-mapping-inhibit*» установлен, то начальное значение равно 0, в противном случае, начальное значение равно *n*+1;

g) переменная «*working_public_key_algorithm*» (действующий алгоритм для открытого ключа) — алгоритм вычисления цифровой подписи, используемый при проверке подписи сертификата. Эта переменная загружается из доверенного алгоритма для открытого ключа, предоставляемого в рамках данных об ДЗУЦ;

h) переменная «*working_public_key*» (действующий открытый ключ) — открытый ключ, используемый для проверки подписи сертификата. Эта переменная загружается из доверенного открытого ключа, предоставляемого в рамках данных об ДЗУЦ;

i) переменная «*working_public_key_parameters*» (параметры действующего открытый ключ) — параметры, касающиеся текущего открытого ключа, которые могут потребоваться для проверки подписи (в зависимости от криптоалгоритма). Эта переменная загружается из параметров доверенного открытого ключа, предоставляемых в рамках данных об ДЗУЦ;

j) переменная «*working_issuer_name*» (действующее имя издателя сертификата) — уникальное имя издателя, ожидаемое в следующем сертификате цепочки сертификатов. Эта переменная загружается из доверенного имени издателя сертификата, предоставляемого в рамках данных об ДЗУЦ;

k) переменная «*max_path_length*» (максимальная длина маршрута) — это целое число, устанавливаемое в значение *n*, уменьшается после обработки каждого не само-изданного сертификата маршрута, и оно может быть уменьшено до значения, указанного в последовательности «*pathLenConstraint*» (ограничение на длину маршрута) субполя «*basicConstraints*» поля «Расширения» сертификата УЦ.

По завершении фазы инициализации, алгоритм переходит в фазу основной обработки сертификата.

§6.1.3 Фаза основной обработки сертификата

В этой фазе обрабатывается *i*-ый сертификат (для всех *i* из [1…*n*]). В это фазе проводятся следующие операции:

a) проверка основной информации сертификата. Сертификат должен удовлетворять, в обязательном порядке, каждому из следующих условий:

1) подпись сертификата может быть проверена с использованием переменных «*working_public_key_algorithm*», «*working_public_key*» и «*working_public_key_parameters*»;

2) период действия сертификата включает текущее время;

3) в текущее время сертификат не является аннулированным. Это может быть установлено путём получения соответствующего СОС, с помощью данных о состоянии или с помощью автономных способов получения информации о сертификате. Имя издателя сертификата соответствует значению переменной «*working_issuer_name*»;

b) если *i*-ый сертификат является само-изданным и не является последним сертификатом маршрута, то пропускаем эту операцию для *i*-ого сертификата. В противном случае, проверяем, что имя владельца сертификата представлено в одном из разрешённых субдеревьев «*permitted_subtrees*» для уникальных имён в соответствие с Рекомендацией ITU-T X.500, а также проверяем, что каждое из альтернативных имён, указанных в субполе «*subjectAltName*» (как критичное или не критичное), представлено в одном из разрешённых субдеревьев «*permitted_subtrees*» для данного формата имён;

c) если *i*-ый сертификат является само-изданным и не является последним сертификатом маршрута, то пропускаем эту операцию для *i*-ого сертификата. В противном случае, проверяем, что имя владельца сертификата не представлено ни в одном из разрешённых субдеревьев «*excluded_subtrees*» для уникальных имён в соответствие с Рекомендацией ITU-T X.500, а также проверяем, что каждое из альтернативных имён, указанных в субполе «*subjectAltName*» (как критичное или не критичное), не представлено ни в одном из разрешённых субдеревьев «*excluded_subtrees*» для данного формата имён;

d) если в сертификате представлено субполе «*certificatePolicies*», а дерево «*valid_policy_tree*» не нулевое (не пустое), то обрабатываем данные о политике следующим образом:

1) для каждой *P*-ой политики, не эквивалентной «*anyPolicy*», представленной в субполе «*certificatePolicies*», пусть *P*-OID обозначает идентификатор *P*-ой политики, а *P*-Q — определитель группы для *P*-ой политики. Проведём следующие операции:

i. для каждого узла глубиной *i*-1 в дереве «*valid_policy_tree*», в котором *P*-OID представляет собой совокупность ожидаемых (вероятных) политик «*expected_policy_set*», сформируем дочерний узел следующим образом: установим «*valid_policy*» в *P*-OID, «*qualifier_set*» *P*-Q, а «*expected_policy_set*» в {*P*-OID}.

Например, рассмотрим дерево «*valid_policy_tree*» с узлом глубиной *i*-1, в котором компонент «*expected_policy_set*» имеет значение {*Gold*, *White*}. Предположим, что в субполе «*certificatePolicies*» *i*-го сертификата представлены политики сертификации «*Gold*» и «*Silver*». Политика «*Gold*» совпала, а политика «*Silver*» — нет. В соответствие с этим правилом будет сформирован дочерний узел глубиной *i* для политики «*Gold*». Результат этой операции представлен на рис. 4;

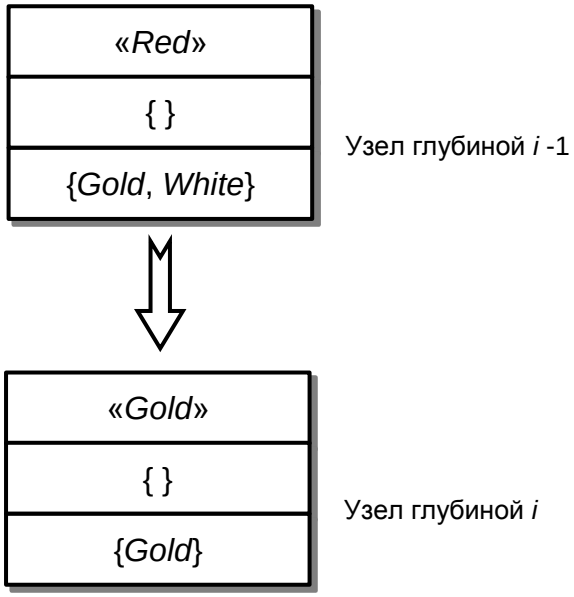

Рис. 4. Операция сравнения на предмет совпадения

ii. Если на *i*-ом шаге совпадение не обнаружено и дерево «*valid_policy_tree*» содержит узел глубиной *i*-1, в котором «*valid_policy*» содержит значение «*anyPolicy*», то формируем дочерний узел со следующими значениями: установим «*valid_policy*» в *P*-OID, «*qualifier_set*» в *P*-Q, а «*expected_policy_set*» в {*P*-OID}.

Например, рассмотрим дерево «*valid_policy_tree*» с узлом глубиной *i*-1, в котором «*valid_policy*» имеет значение «*anyPolicy*». Предположим, что в субполе «*certificatePolicies*» *i*-го сертификата представлены политики сертификации «*Gold*» и «*Silver*». Политика «*Gold*» не содержит определителя, а тоже время политика «*Silver*» имеет определитель «Q-*Silver*». Если политики «*Gold*» и «*Silver*» не совпали, как это было указано выше в §(i), то в соответствие с этим правилом будут сформированы два дочерних узла глубиной *i* для каждой из политик. Результат представлен на рис. 5;

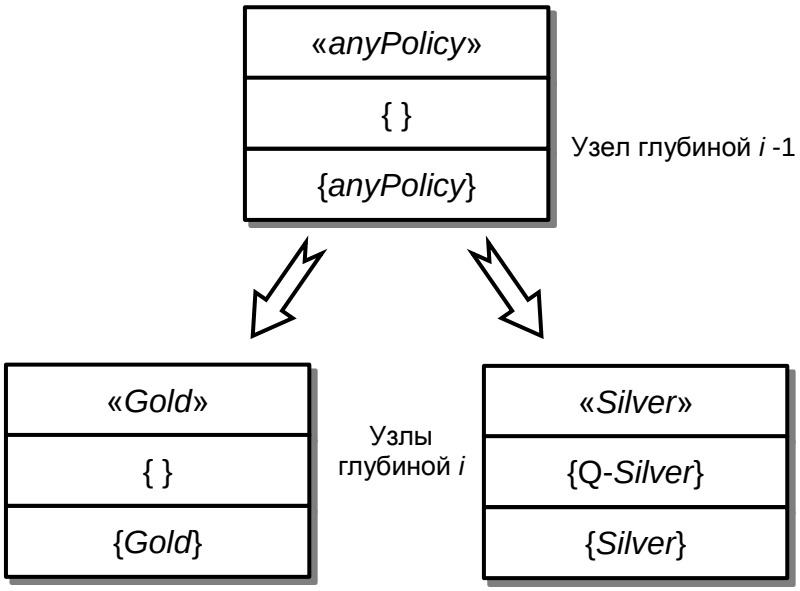

Рис. 5. Операция сравнения, обнаружившая не совпадение политик, когда лист/узел дерева содержит значение «*anyPolicy*»

2) если субполе «*certificatePolicies*» содержит политику «*anyPolicy*» с определителем, установленным в значение «*AP*-Q», а также, (*a*) либо переменная «*inhibit_anyPolicy*» имеет значение > 0, (*b*) либо сертификат является само-изданным, тогда:

Для каждого узла в дереве «*valid_policy_tree*» с узлом глубиной *i*-1, для каждого значения в компоненте «*expected_policy_set*» (включая «*anyPolicy*»), которое отсутствует в дочернем узле, формируем дочерний узел со следующими значениями: установим компонент «*valid_policy*» в значение,

взятое из компонента «*expected_policy_set*» в родительском узле, компонент «*qualifier_set*» — в значение «*AP*-Q», и компонент «*expected_policy_set*» — в значение, взятое из компонента «*valid_policy*» этого узла.

Например, рассмотрим дерево «*valid_policy_tree*» с узлом глубиной *i*-1, в котором компонент «*expected_policy_set*» имеет значение {*Gold, White*}. Предположим, что в субполе «*certificatePolicies*» *i*-го сертификата содержится значение «*anyPolicy*» без определителей политик, но значения «*Gold*» и «*Silver*» отсутствуют. В соответствие с этим правилом будут сформированы два дочерних узла глубиной *i* для каждой из политик. Результат представлен на рис. 6;

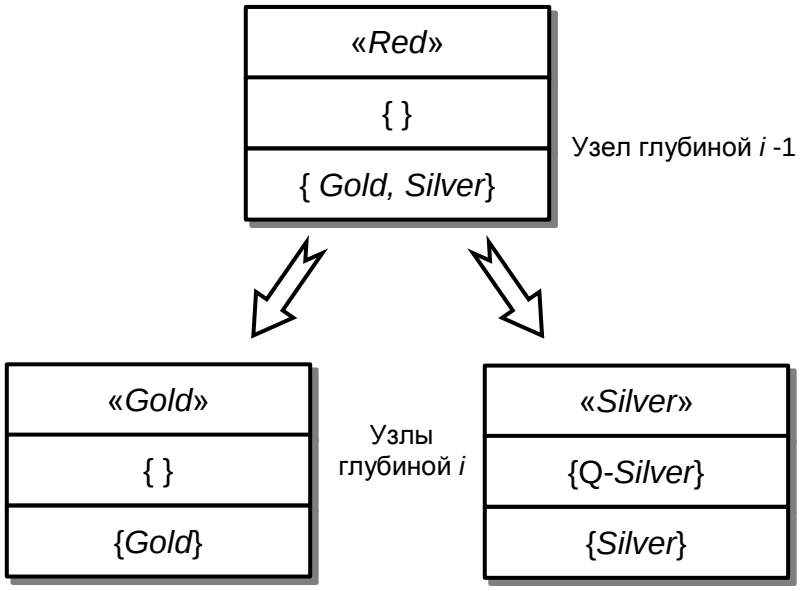

Рис. 6. Операция сравнения, обнаружившая не совпадение политик, когда субполе «*certificatePolicies*» содержит значение «*anyPolicy*»

3) если в дереве «*valid_policy_tree*» с узлом глубиной *i*-1 или меньше присутствует узел без каких-либо дочерних узлов, то удаляем этот узел. Повторяем эту операцию до тех пор, пока не будет узлов глубиной *i*-1 или меньше без дочерних узлов.

Например, рассмотрим дерево «*valid_policy_tree*», изображённое на рис. 7. Два узла глубиной *i*-1, помеченные как «*X*», и не имеющие дочерних узлов, удаляются. Следуя этому правилу, будет сформировано итого-

вое дерево, в котором расположен узел голубиной *i*-2, помеченный как «*Y*» и подлежащий удалению. В итоговом дереве отсутствуют узлы глубиной *i*-1 или меньше, и не имеющие дочерних узлов. И на этом данная операция завершается;

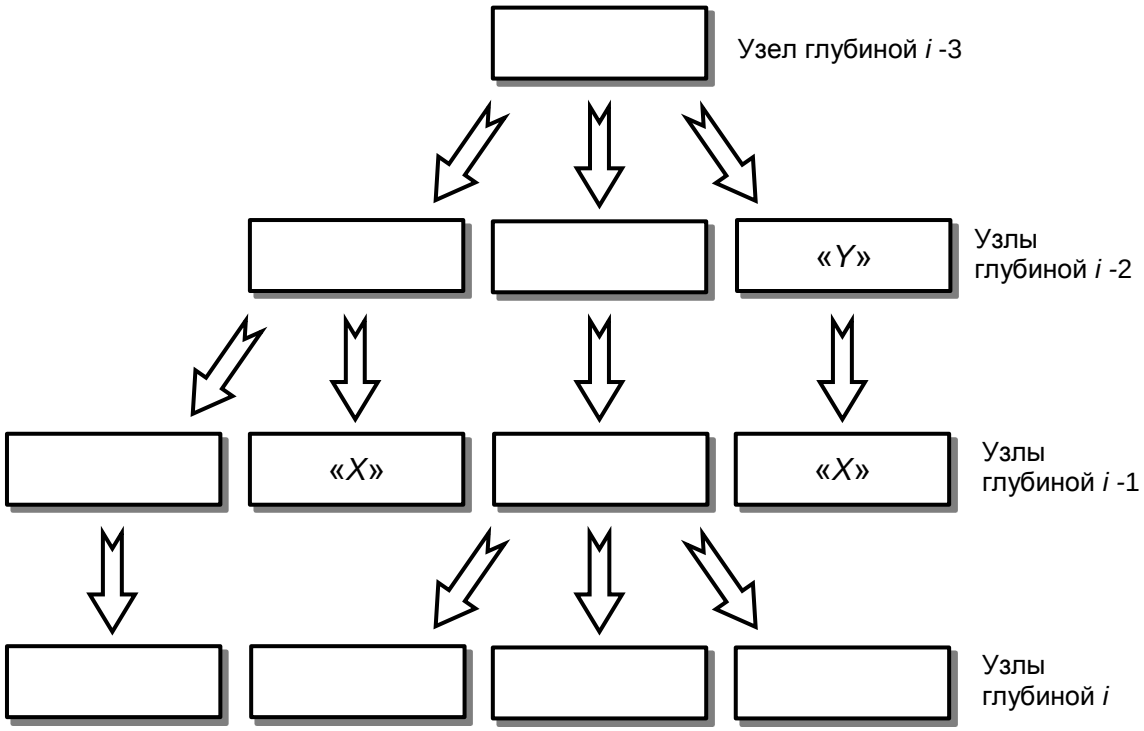

Рис. 7. Упрощение дерева «*valid_policy_tree*»

e) если субполе «*certificatePolicies*» отсутствует, то устанавливаем дерево «*valid_policy_tree*» в нулевое значение («*NULL*»);

f) проверяем, либо переменная «*explicit_policy*» больше нуля, либо переменная «*valid_policy_tree*» не равна нулевому значению («*NULL*»);

Если любая из операций *a), b), c)* или *f)* в фазе инициализации завершилась неудачей, то процедура прекращается, указывая на неисправность и соответствующую причину.

Если *i* не равно *n*, то продолжаем обработку путём реализации подготовительных операций, представленных в §6.1.4. Если *i* равно *n*, то переходим к завершению процедуры (§6.1.5).

§6.1.4 Фаза подготовки к обработке (*i* +1)-го сертификата

Для готовности к обработке (*i*+1)-го сертификата, проводятся следующие операции по отношению к *i*-ому сертификату:

a) если имеет место субполе «*policyMappings*» (отображения политик), то проверяем, что специфическое значение «*anyPolicy*» не представлено в виде последовательности «*issuerDomainPolicy*» или «*subjectDomainPolicy*»;

b) если имеет место субполе «*policyMappings*», то для каждой последовательности «*issuerDomainPolicy*» с идентификатором политики ID-*P* в субполе «*policyMappings*»:

1) если переменная «*policy_mapping*» > 0, то для каждого узла в дереве «*valid_policy_tree*» глубиной *i*, в котором идентификатор политики ID-*P* указывает на приемлемую политику «*valid_policy*», устанавливаем в компоненте «*expected_policy_set*» совокупность значений «*subjectDomainPolicy*», которые определяются как эквиваленты ID-*P* с помощью субполя «*policyMappings*».

Если в дереве «*valid_policy_tree*» нет узла глубиной *i*, имеющего приемлемую политику «*valid_policy*» с идентификатором ID-*P*, но имеет место узел глубиной *i*, имеющий приемлемую политику «*anyPolicy*», то из этого узла формируется дочерний узел глубиной *i*-1 с приемлемой политикой «*anyPolicy*» следующим образом:

i. установим компонент «*valid_policy*» в значение ID-*P*;
ii. установим компонент «*qualifier_set*» в значение совокупности определителей, указанной в политике «*anyPolicy*» из субполя «*certificatePolicies*» *i*-ого сертификата;
iii. и установим компонент «*expected_policy_set*» в совокупность значений «*subjectDomainPolicy*», которые определяются как эквиваленты значения ID-*P* с помощью субполя «*policyMappings*»;

2) если переменная «*policy_mapping*» равна 0:

i. удаляем каждый узел глубиной *i* в дереве «*valid_policy_tree*», в котором ID-*P* определяет приемлемую политику «*valid_policy*»;

ii. если в дереве «*valid_policy_tree*» представлен узел глубиной *i*-1 или меньше без дочерних узлов, то данный узел удаляется. Повторяем эту операцию до тех пор, пока не будет узлов глубиной *i*-1 или меньше без дочерних узлов;

c) присваиваем имя владельца сертификата переменной «*working_issuer_name*»;

d) присваиваем значение «*subjectPublicKey*» из сертификата переменной «*working_public_key*»;

e) если поле «*subjectPublicKeyInfo*» сертификата включает субполе «*algorithm*» с не нулевыми параметрами, то присваиваем эти параметры переменной «*working_public_key_parameters*».

Если же поле «*subjectPublicKeyInfo*» сертификата включает субполе «*algorithm*» с нулевыми параметрами или эти параметры просто отсутствуют, то сравниваем алгоритм для «*subjectPublicKey*» из сертификата с переменной «*working_public_key_algorithm*». Если алгоритм для «*subjectPublicKey*» из сертификата и переменная «*working_public_key_algorithm*» не совпадают, то устанавливаем переменную «*working_public_key_algorithm*» в нулевое значение;

f) присваиваем значение алгоритма для «*subjectPublicKey*» из сертификата переменной «*working_public_key_algorithm*»;

g) если в сертификате представлено субполе «*nameConstraints*», то изменяем переменные состояния «*permitted_subtrees*» и «*excluded_subtrees*» следующим образом:

1) если в сертификате представлена последовательность «*permittedSubtrees*», то устанавливаем переменную состояния «*permitted_subtrees*» в точку пересечения её предыдущего значения со значением, представленном в поле «Расширения». Если последовательность «*permittedSubtrees*» не содержит соответствующий формат имени, то переменная состояния «*permitted_subtrees*» остаётся неизменной для этого типа имён. Например, точкой пересечения имён «*example.com*» и «*foo.example.com*» является «*foo.examp-*

le.com.». А точкой пересечения имён «*example.com*» и «*example.net*» является пустое множество;

2) если в сертификате представлена последовательность «*excludedSubtrees*», то устанавливаем переменную состояния «*excluded_subtrees*» в объединённую последовательность, состоящую из её предшествующего значения и значения, представленного в поле «Расширения». Если последовательность «*excludedSubtrees*» не включает соответствующего формата имён, то переменная состояния «*excluded_subtrees*» остаётся неизменной для этого типа имён. Например, объединённой последовательностью пространств имён «*example.com*» и «*foo.example.com*» является «*example.com.*». А объединённой последовательностью пространств имён «*example.com*» и «*example.net*» являются оба пространства имён;

h) если *i*-ый сертификат не является само-изданным, то:

1) если переменная «*explicit_policy*» ≠ 0, то уменьшаем переменную «*explicit_policy*» на единицу;

2) если переменная «*policy_mapping*» ≠ 0, то уменьшаем переменную «*policy_mapping*» на единицу;

3) если переменная «*inhibit_anyPolicy*» ≠ 0, то уменьшаем переменную «*inhibit_anyPolicy* » на единицу;

i) если сертификат включает субполе «*policyConstraints*», то изменяем переменные состояния «*explicit_policy*» и «*policy_mapping*» следующим образом:

1) если представлена последовательность «*requireExplicitPolicy*» и её значение меньше значения переменной «*explicit_policy*», то устанавливаем переменную «*explicit_policy*» в значение из последовательности «*requireExplicitPolicy*»;

2) если представлена последовательность «*inhibitPolicyMapping*» и её значение меньше значения переменной «*policy_mapping*», то устанавливаем переменную «*policy_mapping*» в значение из последовательности «*inhibitPolicyMapping*»;

j) если сертификат включает субполе «*inhibitAnyPolicy*» и его значение меньше значения переменной «*inhibit_anyPolicy*», то устанавливаем переменную «*inhibit_anyPolicy*» в значение из субполя «*inhibitAnyPolicy*»;

k) если *i*-ый сертификат является третьей версией сертификата, то проверяем, что в этом сертификате представлено субполе «*basicConstraints*», и что флаг «*cA*» установлен в значение «*TRUE*»; (Если *i*-ый сертификат является первой или второй версией сертификата, то прикладная система обязана, либо проверить, что *i*-ый сертификат является сертификатом УЦ, полученный автономным способом, либо удалить этот сертификат. Прикладные системы и ИТС, придерживающиеся этого стандарта, могут удалять все промежуточные сертификата между первой и второй версиями.)

l) если сертификат не был само-изданным, то проверяем, что переменная «*max_path_length*» > 0, и уменьшаем значение переменной «*max_path_length*» на единицу;

m) если сертификат включает последовательность «*pathLenConstraint*» и её значение меньше значения переменной «*max_path_length*», то устанавливаем переменную «*max_path_length*» в значение из последовательности «*pathLenConstraint*»;

n) если сертификат включает субполе «*keyUsage*», то проверяем, бит «*keyCertSign*» установлен;

o) распознаём и обрабатываем любое другое критичное субполе в поле «Расширения», представленное в сертификате. Обрабатываем любое другое распознаваемое не критичное субполе в поле «Расширения», представленное в сертификате, и которое затрагивает обработку маршрута.

Если проверки *a), k), l), n)* или *o)* «провалились», то процедура прекращается, указывая на неисправность и соответствующую причину.

Если проверки *a), k), l), n)* или *o)* завершились успешно, то увеличивает на единицу индекс *i* и продолжаем основную обработку сертификата (§6.1.3).

§6.1.5 Завершающая фаза

Для завершения обработки целевого сертификата, проводим следующие операции относительно *n*-го сертификата:

a) если переменная «*explicit_policy*» ≠ 0, то уменьшаем переменную «*explicit_policy*» на единицу;

b) если сертификат включает субполе «*policyConstraints*» и представлена последовательность «*requireExplicitPolicy*», имеющая нулевое значение, то устанавливаем переменную состояния «*explicit_policy*» в нулевое значение;

c) присваиваем значение «*subjectPublicKey*» из сертификата переменной «*working_public_key*»;

d) если поле «*subjectPublicKeyInfo*» сертификата включает субполе «*algorithm*» с не нулевыми параметрами, то присваиваем эти параметры переменной «*working_public_key_parameters*».

Если же поле «*subjectPublicKeyInfo*» сертификата включает субполе «*algorithm*» с нулевыми параметрами или эти параметры просто отсутствуют, то сравниваем алгоритм для «*subjectPublicKey*» из сертификата с переменной «*working_public_key_algorithm*». Если алгоритм для «*subjectPublicKey*» из сертификата и переменная «*working_public_key_algorithm*» не совпадают, то устанавливаем переменную «*working_public_key_algorithm*» в нулевое значение;

e) присваиваем значение алгоритма для «*subjectPublicKey*» из сертификата переменной «*working_public_key_algorithm*»;

f) распознаём и обрабатываем любое другое критичное субполе в поле «Расширения», представленное в *n*-ом сертификате. Обрабатываем любое другое распознаваемое не критичное субполе в поле «Расширения», представленное в *n*-ом сертификате, и которое затрагивает обработку маршрута.

g) рассчитываем точку пересечения дерева «*valid_policy_tree*» и параметра «*user-initial-policy-set*» следующим образом:

i. если дерево «*valid_policy_tree*» = 0, то точка пересечения = 0;

ii. если дерево «*valid_policy_tree*» ≠ 0 и параметр «*user-initial-policy-set*» имеет значение «*any-policy*», то точка пересечения является всё дерево «*valid_policy_tree*»;

iii. если дерево «*valid_policy_tree*» ≠ 0 и параметр «*user-initial-policy-set*» не имеет значение «*any-policy*», то рассчитываем точку пересечения дерева «*valid_policy_tree*» и параметра «*user-initial-policy-set*» следующим образом:

1. определяем совокупность узлов, определяющих политики, чьи родительские узлы имеют переменные «*valid_policy*» со значением «*anyPolicy*». Это и есть совокупность «*valid_policy_node_set*»;

2. если компонент «*valid_policy*» любого узла из совокупности «*valid_policy_node_set*» не представлен в параметре «*user-initial-policy-set*» и не имеет значения «*anyPolicy*», то удаляем этот узел и все его дочерние узлы;

3. если дерево «*valid_policy_tree*» включает узел глубиной *n* с компонентом «*valid_policy*», имеющим значение «*anyPolicy*», а параметр «*user-initial-policy-set*» не содержит значения «*anyPolicy*», то выполняем следующие операции:

a. установим компонент «*qualifier_set*» узла глубиной *n* с компонентом «*valid_policy*», имеющим значение «*anyPolicy*», в значение *P*-Q;

b. для каждого идентификатора *P*-OID в параметре «*user-initial-policy-set*», который не является приемлемой политикой «*valid_policy*» узла в совокупности «*valid_policy_node_set*», формируем дочерний узел, у которого существует родительский узел глубиной *n*-1 с приемлемой политикой «*valid_policy*», содержащей значение «*anyPolicy*». Устанавливаем значения в дочернем узле следующем образом: установим «*valid_policy*» в *P*-OID, «*qualifier_set*» в *P*-Q, а «*expected_policy_set*» в {*P*-OID};

c. удаляем узел глубиной *n* с приемлемой политикой «*valid_policy*», содержащей значение «*anyPolicy*»;

4. если в дереве «*valid_policy_tree*» существует узел глубиной *n*-1 или меньше и без каких-либо дочерних узлов, то удаляем этот узел.

Повторяем эту операцию до тех пор, пока не останется узлов глубиной *n*-1 или меньше и без каких-либо дочерних узлов.

Если, либо (1) значение переменной «*explicit_policy*» > 0, либо (2) дерево «*valid_policy_tree*» не нулевое (не пустое), то обработка маршрута завершилась успешно.

§6.1.6 Выходные данные

Если обработка МС завершилась успешно, ПрПП завершается и отображается её результат, а также заключительные значения переменных величин «*valid_policy_tree*», «*working_public_key*», «*working_public_key_algorithm*» и «*working_public_key_parameters*».

§6.2 Использование алгоритма ПрПП маршрута

Алгоритм ПрПП маршрута сертификации описывает операции обработки одиночного МС. Несмотря на то, что каждый МС начинается с ДЗУЦ, не существует требований, чтобы все МС, проверяемые на подлинность с помощью соответствующей системы, использовали совместно только одиночный ДЗУЦ. Выбор одного или более доверенных УЦ является локальной задачей. Система может назначать любой УЦ из числа доверенных в качестве ДЗУЦ для конкретного МС. Входные данные алгоритма ПрПП могут быть разными в зависимости от МС. Входные данные, используемые при обработке маршрута, могут отражать специфику требований прикладной системы или ИТС, а также ограничений, накладываемых на доверие в интересах реализации соответствующей политики сертификации. Например, доверенный УЦ может иметь статус доверенного только в интересах реализации соответствующей политики сертификации. Такое ограничение может быть реализовано с помощью используемых для ПрПП входных данных.

Прикладные системы и ИТС могут дополнить алгоритм, представленный в §6.1, с целью дополнительного ограничения совокупности приемлемых МС, которые начинаются с соответствующего ДЗУЦ. Например, прикладные систе-

мы и ИТС могут модифицировать алгоритм с целью использования последовательности «*pathLenConstraint*» (ограничение на длину маршрута) субполя «*basicConstraints*» в поле «Расширения» сертификата для конкретного ДЗУЦ в течении фазы инициализации, либо прикладная система может потребовать наличие соответствующего формата альтернативного имени в целевом сертификате, либо прикладная система может установить требования для конкретных прикладных расширений. Таким образом, алгоритм ПрПП, представленный в §6.1, устанавливает минимально необходимые условия для МС, который считается приемлемым.

Там, где УЦ распространяет само-подписанный сертификаты в целях определения информации о ДЗУЦ, для описания рекомендованных входных данных для ПрПП может использоваться поле «Расширения» сертификата. Например, субполе «*policyConstraints*» поля «Расширения» могло бы использоваться в само-подписанном сертификате для указания того, что МС, начинающиеся с этого ДЗУЦ, должны быть надёжными только для конкретных политик. Аналогично, субполе «*nameConstraints*» поля «Расширения» могло бы указывать на то, что МС, начинающиеся с этого ДЗУЦ, должны быть надёжными только для конкретных пространств имён. Алгоритм ПрПП маршрута, представленный в §6.1, не предполагает, что данные о ДЗУЦ предоставляются в само-подписанных сертификатах, и не устанавливает правил обработки дополнительной информации, представленной в таких сертификатах. Тем не менее, стандарт RFC-5914 устанавливает несколько форматов для кодирования информации о ДЗУЦ, включая само-подписанные сертификаты, а стандарт RFC-5937 приводит пример того, как такая информация может использоваться в качестве входных данных в фазе инициализации ПрПП маршрута. Прикладные системы и ИТС свободны в выборе варианта использования любой дополнительной информации, которая включена в структуру данных о ДЗУЦ, либо они в праве проигнорировать такую информацию.

§6.3 ПрПП списков отозванных сертификатов

Далее рассматриваются операции, необходимые для определения, является ли сертификат отозванным в тех случаях, когда списки отозванных сертификатов являются способом аннулирования, используемым издателем сертификатов. От прикладных систем и ИТС, придерживающихся данного стандарта и обрабатывающие СОС, не требуется применять данный алгоритм, но они обязаны использовать эквивалентную ПрПП СОС, обеспечивающую функциональную совместимость с внешней процедурной характеристикой, реализуемой X.509-алгоритмом обработки СОС, которые были изданы в соответствие с данным стандартом. Соответствующая прикладная система или ИТС может использовать любой алгоритм сколь угодно долго, но при условии, что он будет приводить к корректному результату. Данный алгоритм подразумевает, что все необходимые СОС, хранящиеся в локальной сверхоперативной памяти (*local cache*), являются доступными. Более того, если время очередного обновления данных СОС прошло, то алгоритм самостоятельно реализует способ получения текущего (действующего) СОС и размещения его в локальной кэш-памяти, предназначенной для хранения СОС.

Этот алгоритм устанавливает совокупности входных данных, переменных состояния и операции обработки, которые используются при обработке каждого сертификата в МС.

§6.3.1 Входные данные для аннулирования

Для обеспечения процедуры аннулирования алгоритму требуются два вида входных данных:

а) сертификат: для алгоритма необходимы последовательный номер сертификата и имя издателя с целью определения, представлен ли сертификат в соответствующем СОС. Субполе «*basicConstraints*» в поле «Расширения» используется для определения, связан ли обрабатываемый сертификат с УЦ или конечным пользователем. Если представлено, то алгоритм использует субполя «*cRLDistributionPoints*» и «*freshestCRL*» для определения состояния отзыва;

b) использование усечённых СОС («*use-deltas*»): это входное булево число, которое определяет, используются для СОС усечённые СОС.

§6.3.2 Фаза инициализации и переменные состояния отзыва

Для обеспечения процедуры обработки СОС алгоритму требуются следующие переменные состояния:

a) маска причин («*reasons_mask*»). Эта переменная включает набор причин отзыва, содержащихся в СОС и усечённых СОС, которые обрабатываются в дальнейшем. В этот набор могут входить только приемлемые причины отзыва, т.е. все возможные, за исключением не установленных, а именно: «*keyCompromise*», «*cACompromise*», «*affiliationChanged*», «*superseded*», «*cessationOfOperation*», «*certificateHold*», «*privilegeWithdrawn*» и «*aACompromise*». Существует специальное значение «*all-reasons*», которое указывает на то, что в набор входят все приемлемые причины отзыва. Эта переменная в фазе инициализации является пустым набором;

b) состояние сертификата («*cert_status*»). Эта переменная содержит указатель состояния сертификата. Ей может быть присвоено одно из следующих значений: «*unspecified*», «*keyCompromise*», «*cACompromise*», «*affiliationChanged*», «*superseded*», «*cessationOfOperation*», «*certificateHold*», «*removeFromCRL*», «*privilegeWithdrawn*», «*aACompromise*», специальное значение «*UNREVOKED*» или специальное значение «*UNDETERMINED*». Эта переменная в фазе инициализации имеет специальное значение «*UNREVOKED*»;

c) промежуточная маска причин («*interim_reasons_mask*»). Эта переменная включает набор причин отзыва, содержащихся в СОС и усечённых СОС, которые находятся в обработке в текущий момент времени.

Следует отметить, что в некоторых прикладных системах и ИТС не необходимости проверять все коды причин отзыва. Например, некоторые системы интересуют только сертификаты УЦ с кодами причин «*cACompromise*» и «*keyCompromise*». Данный алгоритм проверяет все коды причин отзыва. Для огра-

ниченной проверки подмножества кодов причин отзыва может понадобиться дополнительные проверка и переменные состояния.

§6.3.3 Обработка COC

Алгоритм обработки начинается с предположения, что сертификат не является аннулированным. Алгоритм проверяет один или несколько COC до тех пор, пока, либо не будет определён состояние сертификата, подлежащего аннулированию, либо будет проверено достаточное число COC с целью выявления всех кодов причин отзыва.

Для каждой точки распространения, указанной в субполе «*cRLDistributionPoints*» сертификата, для каждого соответствующего COC, хранящегося в локальной кэш-памяти для COC, несмотря на то, что переменная «*reasons_mask*» не содержит значение «*all-reasons*», а переменная «*cert_status*» — «*UNREVOKED*», проводим следующие операции:

a) обновляем данные в локальной кэш-памяти для COC за счёт получения заполняемого COC, усечённого COC или их обоих, если конечно это потребуется:

1) если текущее время больше времени, указанного в субполе «*nextUpdate*» COC, то выполняем одну из двух следующих операций:

i. если значение «*use-deltas*» установлено, а сертификат или COC содержат субполе «*freshestCRL*», то получаем усечённый COC со следующим значением в субполе «*nextUpdate*», которое позднее текущего времени, и может использоваться для обновления данных в локальной кэш-памяти для COC (§5.2.4);

ii. добавляем заполняемый COC к данным в локальной кэш-памяти для COC, проверяем, что текущее время более раннее, чем значение в субполе «*nextUpdate*» нового COC, и продолжаем обработку с использование нового COC. Если значение «*use-deltas*» установлено, а сертификат или COC содержат субполе «*freshestCRL*», то получаем текущий усечённый COC, который может использоваться для обновления ново-

го заполняемого СОС, хранящегося в локальной кэш-памяти для СОС (§5.2.4);

4) если текущее время больше значения в субполе «*nextUpdate*», значение «*use-deltas*» установлено, а сертификат или СОС содержат субполе «*freshestCRL*», то получаем текущий усечённый СОС, который может использоваться для обновления заполняемого СОС, хранящегося в локальной кэш-памяти для СОС (§5.2.4);

b) проверяем издателя и область применения заполняемого СОС следующим образом:

1) если точка распространения включает субпоследовательность «*cRLIssuer*», то проверяем, что поле «*Issuer*» в заполняемом СОС совпадает с субпоследовательностью «*cRLIssuer*» в точке распространения, и что заполняемый СОС содержит последовательность «*issuingDistributionPoint*», которая, в свою очередь, включает субпоследовательность «*indirectCRL*» (содержит логический тип данных, «*boolean*»). В противном случае, проверяем, что издатель СОС совпадает с издателем сертификата;

2) если заполняемый СОС включает последовательность «*issuingDistributionPoint*» субполя «*crlExtensions*», то проверяем:

i. если в последовательности «*issuingDistributionPoint*» субполя «*crlExtensions*» представлена субпоследовательность «*DistributionPointName*», а в последовательности «*DistributionPoint*» представлена субпоследовательность «*Distribution*», то проверяем, что одно из наименований в последовательности «*issuingDistributionPoint*» совпадает с одним из наименований в последовательности «*DistributionPoint*». Если в последовательности «*issuingDistributionPoint*» субполя «*crlExtensions*» представлена субпоследовательность «*DistributionPointName*», а в последовательности «*DistributionPoint*» не представлена субпоследовательность «*Distribution*», то проверяем, что одно из наименований в последовательности «*issuingDistributionPoint*» совпа-

дает с одним из наименований в субпоследовательности «*cRLIssuer*» последовательности «*DistributionPoint*»;

ii. если в последовательности «*issuingDistributionPoint*» субполя «*crlExtensions*» представлена субпоследовательность «*onlyContainsUserCerts*» (содержит логический тип данных, «*boolean*»), то проверяем, что в субполе «*basicConstraints*» поля «Расширения» сертификата не установлен флаг «*cA*»;

iii. если в последовательности «*issuingDistributionPoint*» субполя «*crlExtensions*» представлена субпоследовательность «*onlyContainsCACerts*» (содержит логический тип данных, «*boolean*»), то проверяем, что в субполе «*basicConstraints*» поля «Расширения» сертификата установлен флаг «*cA*»;

iv. проверяем, что субпоследовательность «*onlyContainsAttributeCerts*» (содержит логический тип данных, «*boolean*») не представлена;

c) если значение «*use-deltas*» установлено, то проверяем издателя и область применения усечённого СОС следующим образом:

1) проверяем, что издатель усечённого СОС совпадает с издателем заполняемого СОС;

2) если заполняемый СОС включает последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*», то проверяем, что усечённый СОС содержит аналогичную (полностью совпадающую) последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*». Если же заполняемый СОС не включает последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*», то проверяем, что и усечённый СОС также не содержит последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*»;

3) проверяем, что последовательность «*authorityKeyIdentifier*» усечённого СОС совпадает с последовательностью «*authorityKeyIdentifier*» заполняемого СОС;

d) вычисляем переменную «*interim_reasons_mask*» для данного СОС следующим образом:

1) если последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*» представлена и она включает субпоследовательность «*onlySomeReasons*», а также последовательность «*DistributionPoint*» содержит субпоследовательность «*reasons*», тогда присваиваем переменной «*interim_reasons_mask*» значение в точке пересечения причин из последовательности «*DistributionPoint*» и из субпоследовательности «*onlySomeReasons*» последовательности «*issuingDistributionPoint*»;

2) если последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*» включает субпоследовательность «*onlySomeReasons*», а последовательность «*DistributionPoint*» не содержит субпоследовательность «*reasons*», тогда присваиваем переменной «*interim_reasons_mask*» значение, содержащееся в субпоследовательности «*onlySomeReasons*» последовательности «*issuingDistributionPoint*»;

3) если последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*» не представлена или она не включает субпоследовательность «*onlySomeReasons*», но последовательность «*DistributionPoint*» содержит субпоследовательность «*reasons*», тогда присваиваем переменной «*interim_reasons_mask*» значение, содержащееся в субпоследовательности «*reasons*» последовательности «*DistributionPoint*»;

4) если последовательность «*issuingDistributionPoint*» в субполе «*crlExtensions*» не представлена или она не включает субпоследовательность «*onlySomeReasons*» и последовательность «*DistributionPoint*» не содержит субпоследовательность «*reasons*», тогда присваиваем переменной «*interim_reasons_mask*» специальное значение «*all-reasons*»;

e) проверяем, что переменная «*interim_reasons_mask*» включает одну или несколько причин, которые не содержатся в переменной «*reasons_mask*»;

f) получаем и подтверждаем подлинность маршрута сертификации до издателя заполняемого СОС. ДЗУЦ для МС должен быть, в обязательном порядке, тем же самым, как и ДЗУЦ, используемы при подтверждении подлинности

целевого сертификата. Если в сертификате издателя СОС представлено субполе «*keyUsage*», то проверяем, что флаг «*cRLSign*» установлен;

g) подтверждаем подлинность подписи в заполняемом СОС, используя для этого открытый ключ, подлинность которого была подтверждена в операции *f)*;

h) если значение «*use-deltas*» установлено, то подтверждаем подлинность подписи в усечённом СОС, используя для этого открытый ключ, подлинность которого была подтверждена в операции *f)*;

i) если значение «*use-deltas*» установлено, то осуществляем поиск сертификата в усечённом СОС. Если запись найдена и она совпадает с издателем сертификата и его серийным номером (§5.3.3), то присваиваем значение переменной «*cert_status*» с указанием причины следующим образом:

1) если код причины отзыва в расширении записи СОС представлен, то присваиваем переменной «*cert_status*» значение этого кода причины отзыва из расширения записи СОС;

2) если код причины отзыва в расширении записи СОС не представлен, то присваиваем переменной «*cert_status*» значение «*unspecified*»;

j) если переменная «*cert_status*» имеет значение «*UNREVOKED*», то осуществляем поиск сертификата в заполняемом СОС. Если запись найдена и она совпадает с издателем сертификата и его серийным номером (§5.3.3), то присваиваем значение переменной «*cert_status*» с указанием причины, как это было сделано в операции *i)*;

k) если переменная «*cert_status*» имеет значение «*removeFromCRL*», то присваиваем переменной «*cert_status*» значение «*UNREVOKED*»;

l) присваиваем переменной состояния «*reasons_mask*» объединённое значение, состоящее из её предшествующего значения и значения переменной состояния «*interim_reasons_mask*».

Если переменная «*reasons_mask*» имеет специальное значение «*all-reasons*» или переменная «*cert_status*» не имеет значения «*UNREVOKED*», то

считаем, что состояние отзыва было установлено, то возвращаемся к переменной «*cert_status*».

Если же состояние отзыва не было установлено, то повторяем предшествующую процедуру с любыми приемлемыми СОС, которые не были указаны в последовательности «*DistributionPoint*», но были выпущены издателем сертификата. При обработке такого СОС полагаем, что последовательность «*DistributionPoint*» не содержит субпоследовательностей «*reasons*» и «*cRLIssuer*», а представлена субпоследовательность «*DistributionPointName*» с именем издателя сертификата (поле «*Issuer*»). Т.е. последовательность имён в полном наименовании «*fullName*» сформирована на основании поля «*Issuer*» сертификата, а также субполя «*issuerAltName*» в поле «*Расширения*» сертификата. После обработки таких СОС, если по-прежнему состояния отзыва не установлено, то присваиваем переменной «*cert_status*» значение «*UNDETERMINED*».

VII ПРАВИЛА ОБРАБОТКИ НАИМЕНОВАНИЙ, ОСНОВАННЫХ НА НАЦИОНАЛЬНЫХ АЛФАВИТАХ

Имена (наименования), основанные на национальных алфавитах (ИНА, *internationalized names*), могут вызвать затруднения при их обработке. Тем не менее, ИНА встречаются во многих сертификатах и полях СОС, а также в их расширениях, включая уникальные имена, имена сегментов/областей, основанные на национальных алфавитах (ИСНА, *internationalized domain names*), адреса электронной почтовой службы, идентификаторы ресурсов, основанные на национальных алфавитах (ИРНА, *internationalized resource identifiers*). Хранение, сравнение, представление и иная обработка таких имён требует особой тщательности и осторожности. Это связано с тем, что некоторые символы могут кодироваться различными способами. Более того, одни и те же имена могут быть представлены в нескольких кодировках (например, *ASCII*- или *UTF8*-код). Далее рассматривается согласование требований для хранения или сравнения каждого из этих форматов имён. Для отдельных форматов имён приводятся правила их отображения.

§7.1 Отображение ИНА в уникальные имена

Отображение ИНА в уникальные имена представлено в §4.1.2.4 для имени издателя сертификата «*Issuer*», в §4.1.2.6 для имени владельца сертификата «*Subject*». Стандартные атрибуты именования, например, общепринятое имя «*common name*», применяют формат данных «*DirectoryString*», который согласуется с ИНА за счёт использования многообразия языковых кодов. Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны обрабатывать *UTF8String*- и *PrintableString*-кодировки. Стандарт RFC-3280 требовал только двоичного сравнения значений атрибутов, закодированных с помощью *UTF8String*-кода, однако, данный стандарт требует более комплексной процедуры сравнения. Прикладные системы могут встретить сертификаты и СОС с именами, которые имеют формат *TeletexString*-, *BMPString*- или *UniversalString*-кодировок, в то время как они рассматривают их как не обязательные («*OPTIONAL*»).

Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны использовать описание «*StringPrep*» LDAP-протокола (включая обработку незначительного множества, RFC-4518) в качестве основы для сравнения атрибутов уникальных имён, имеющих формат, либо *UTF8String*-, либо *PrintableString*-кодировки. Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны проводить процедуры сравнения имён с использованием правила сравнения «*caseIgnoreMatch*» (Рекомендация ITU-T X.520). Применение атрибутов различных форматов, предусматривающих иные подобные правила сравнения, является не обязательным.

Перед тем, как проводить сравнение имён на основе правила сравнения «*caseIgnoreMatch*», прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны реализовать шести итерационную процедуру (алгоритм) подготовки последовательности, представленную в стандарте RFC-4518, для каждого атрибута в формате «*DirectoryString*», с учётом следующих изменений:

∗ на второй итерации, «*Map*» (отображение/преобразование), целесообразно, чтобы преобразование включало игнорирование регистра написания символов («*case folding*»), как это определено в Приложении B.2 стандарта RFC-3454;

∗ на шестой итерации, «*Insignificant Character Removal*» (удаление незначащего символа), проводится компрессия за счёт удаления символов «пробел», «табуляция» и «пустая строка», как это определено в §2.6.1 стандарта RFC-4518 (обработка пустых символов, «*Insignificant Space Handling*»).

При подготовке последовательности символов, атрибуты должны, в обязательном порядке, трактоваться как сохраняемые значения.

Сравнения атрибутов «*domainComponent*» должно проводиться, в обязательном порядке, как это описано в §7.3. Считается, что два атрибута именования совпадают, если типы атрибутов совпали, а значения атрибутов полностью совпали после выполнения процедуры подготовки последовательности символов. Два взаимосвязанных уникальных имени *RDN1* и *RDN2* совпадают, если они имеют одно и то же число атрибутов именования и для каждого такого атрибута в имени *RDN1* существует точно такой же (полностью совпадающий) атрибут именования в имени *RDN2*. Два уникальных наименования *DN1* и *DN2* совпадают, если они имеют одно и то же число взаимосвязанных уникальных имён *RDN*, для каждого *RDN* в *DN1* существует точно такое же (полностью совпадающее) *RDN* в *DN2* и все совпадающие *RDN* расположены в обоих уникальных наименованиях *DN* в одном и том же порядке. Уникальное наименование *DN1* расположено в границах субдерева, устанавливаемого уникальным наименованием *DN2*, если *DN1* содержит, по крайней мере, столько же взаимосвязанных уникальных имён *RDN*, как и в *DN2*, а сами *DN1* и *DN2* совпадают, если замыкающие *RDN* в *DN1* игнорируются.

§7.2 Отображение ИСНА в обобщённые имена «GeneralName»

ИСНА могут содержаться в сертификатах и СОС, а именно в субполях «*subjectAltName*», «*issuerAltName*», «*nameConstraints*», «*subjectInfoAccess*», «*au-*

thorityInfoAccess» и «*cRLDistributionPoints*» поля «*Расширения*» сертификата, в последовательностях «*authorityInformationAccess*» и «*issuingDistributionPoint*» субполя «*crlExtensions*» («*Расширения COC*»). Каждое из этих расширений использует формат обобщённых имён «*GeneralName*». Единственным вариантом обобщённого имени «*GeneralName*» является *dNSName*-формат с соответствующей кодировкой «*IA5String*».

Кодировка «*IA5String*» ограничивается совокупностью *ASCII*-символов. Для адаптации ИСНА к текущему формату, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны преобразовывать ИСНА, перед тем как сохранить его в поле «*dNSName*», в формат кодирования, совместимый с *ASCII*-кодом (*ASCII Compatible Encoding* — ACE), как это определено в §4 стандарта RFC-3490. А именно, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны провести процедуру преобразования, как это определено в §4 стандарта RFC-3490, с учётом следующих изменений:

∗ на первой итерации, целесообразно, чтобы наименование сегмента/области рассматривалось, как «сохраняемая последовательность символов» («*stored string*»). Т.е. флаг «*AllowUnassigned*» не должен быть установлен;

∗ на третьей итерации, устанавливаем флаг, именуемый как «*UseSTD3-ASCIIRules*»;

∗ на четвёртой итерации, обрабатываем каждый маркер с помощью операции «*ToASCII*»;

∗ и, на пятой итерации, изменяем все разделители в последовательности маркеров на код «*U+002E*» (далее полная остановка).

Когда сравниваются DNS-имена на предмет их совпадения, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны провести процедуру точного сравнения всего DNS-имени в режиме игнорирования регистра написания символов. Когда анализируется субполе «*nameConstraints*», прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны провести процедуру точного последовательного сравнения маркеров (*label-by-label basis*) в режиме игнорирования регистра написания символов. Как было отме-

чено в §4.2.1.10, любое DNS-имя, которое может быть сформировано путём добавления маркеров с левой стороны наименования сегмента/области, являющегося запрещённым, считается неприемлемым в рамках указанного субдерева.

Целесообразно, чтобы прикладные системы и ИТС отображали ИСНА в Юникод (*Unicode*)♦, перед тем, как выводить ИСНА на дисплей. А именно, целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, проводили процедуру преобразования, представленную в §4 стандарта RFC-3490, с учётом следующих изменений:

∗ на первой итерации, целесообразно, чтобы наименование сегмента/области рассматривалось, как «сохраняемая последовательность символов». Т.е. флаг «*AllowUnassigned*» не должен быть установлен;

∗ на третьей итерации, устанавливаем флаг, именуемый как «*UseSTD3-ASCIIRules*»;

∗ на четвёртой итерации, обрабатываем каждый маркер с помощью операции «*ToASCII*»;

∗ переход на пятую итерацию.

(Примечание. Прикладные системы и ИТС обязаны признавать расширение диапазона требований в интересах ИСНА. ACE-маркер в ИСНА будет начинаться с четырёх дополнительных символов «xn--» и может потребовать и более пяти ASCII-символов с целью указания одиночного международного символа.)

§7.3 Отображение ИСНА в уникальные имена

Наименования сетевых сегментов/областей могут быть также представлены в формате уникальных имён, используя компоненты сетевого сегмента/области, содержащихся в полях «*Subject*», «*Issuer*» и субполях «*subjectAltName*» и «*issuerAltName*» поля «Расширения». Как и в случае наименования сетевого сегмента/области «*dNSName*» в формате «*GeneralName*», значение этого атрибута кодируется как «*IA5String*». Каждый атрибут «*domainComponent*»

♦ 16-битный стандарт кодирования символов, позволяющий представлять алфавиты всех существующих в мире языков (ISO/IEC 9899:2007).

представляется в виде одиночного маркера. Для отображения маркера из ИСНА в уникальной имя, прикладные системы и ИТС обязаны преобразовать маркер с помощью операции «*ToASCII*», представленной в §4.1 стандарта RFC-3490, с одновременной установкой флага «*UseSTD3ASCIIRules*». Целесообразно, чтобы маркер рассматривался как «сохраняемая последовательность символов». Т.е. флаг «*AllowUnassigned*» не должен быть установлен. Процедура преобразования аналогично той, которая представлена в четвёртой итерации §7.2. Целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, проводили процедуру точного сравнения атрибутов «*domainComponent*» в режиме игнорирования регистра написания символов, как это описано в §7.2.

Целесообразно, чтобы прикладные системы и ИТС отображали ACE-маркеры в Юникод (*Unicode*), перед тем, как выводить ACE-маркеры на дисплей. В частности, целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, проводили операцию отображения «*ToASCII*», представленную в §7.2, над каждым ACE-маркером, перед тем как на экран будет выведено имя.

§7.4 Отображение ИРНА

ИРНА-идентификаторы представляют собой дополнение, основанное на национальных алфавитах, к URI-идентификатору. ИРНА-идентификаторы — это последовательности символов Юникода, в то время как URI-идентификаторы — это последовательность символов *ASCII*-кода. Стандарт RFC-3987 устанавливает правила отображения ИРНА-идентификаторов в URI-идентификаторы. Несмотря на то, что ИРНА-идентификаторы не кодируются непосредственно в полях или расширениях сертификатов, являющиеся их отображением URI-идентификаторы могут включаться в сертификаты и СОС. URI-идентификаторы могут быть включены в субполя «*subjectAltName*», «*issuerAltName*», «*nameConstraints*», «*authorityInfoAccess*», «*subjectInfoAccess*» и «*cRLDistributionPoints*» поля «Расширения» сертификата, последовательности «*authorityInformationAccess*» и «*issuingDistributionPoint*» субполя «*crlExtensions*» («*Расшире-*

ния COC»). Каждое из этих расширений использует формат обобщённых имён «*GeneralName*». URI-идентификаторы размещаются в последовательности «*uniformResourceIdentifier*» в формате «*GeneralName*», который кодируется как «*IA5String*».

Для включения ИРНА-идентификаторов в рассматриваемую структуру, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны преобразовывать ИРНА-идентификаторы в URI-идентификаторы, как это определено в §3.1 стандарта RFC-3987, с учётом следующих изменений:

∗ на первой итерации, формируем последовательность UCS-символов[°] из оригинального ИРНА-формата в соответствие с NFC-формой, как это определено Вариантом «*b*» в «Форматах нормирования Юникода»[•];

∗ выполняем вторую итерацию, использую выходные данные первой итерации.

Прикладные системы и ИТС обязаны не преобразовывать компонент «*ireg-name*» до выполнения второй итерации.

Прежде чем URI-идентификаторы будут сравниваться, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны провести комбинацию процедур по их нормированию на основе правил синтаксиса и схемы построения, которые установлены стандартом RFC-3987. А именно, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны подготовить URI-идентификаторы для их сравнения следующим образом:

∗ первая итерация. Там, где ИРНА-идентификаторы позволяют использовать ИСНА, следует, в обязательном порядке, преобразовать такие наименования в последовательность ACE-маркеров, как это определено в §7.2;

∗ вторая итерация. Схема построения и наименование IP-узла нормируются в нижний регистр, как это определено в §5.3.2.1 стандарта RFC-3987;

[°] Стандарт ISO 10646: Совокупность универсальных много-октетных кодовых символов (Universal multiple-octet coded Character Set, UCS).

[•] Davis M. and Duerst M., «Unicode Standard Annex #15: Unicode Normalization Forms», October 2006, http://www.unicode.org/reports/tr15/.

∗ третья итерация. Производим перекодировку октетов в шестнадцатеричный код, как это определено в §5.3.2.3 стандарта RFC-3987;

∗ четвёртая итерация. Производим нормирование (удаление) сегмента маршрута, разделённого точками («.» и «..»), как это определено в §5.3.2.3 стандарта RFC-3987;

∗ пятая итерация. Если URI-идентификаторы установлены (определены их схемы построения), то прикладной процесс обязан провести их нормирование по правилам схемы построения таких идентификаторов, как это определено в §5.3.3 стандарта RFC-3987.

Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны распознавать и нормировать на основе правил построения схемы идентификации следующие схемы: «*LDAP*», «*HTTP*», «*HTTPS*» и «*FTP*». Если схема не распознана, то пятая итерация исключается.

Если проводится сравнение эквивалентных URI-идентификаторов, то целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, осуществляли сравнение в режиме игнорирования регистра написания символов.

Целесообразно, чтобы прикладные системы и ИТС осуществляли преобразование URI-идентификаторов в Юникод, перед их отображением на дисплее. В частности, целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, осуществляли такое преобразование в соответствие с правилами, представленными в §3.2 стандарта RFC-3987.

§7.5 Отображение адресов электронной почтовой службы, основанных на национальных алфавитах

Адреса почтовой службы могут содержаться в сертификатах и СОС, а именно в субполях «*subjectAltName*», «*issuerAltName*», «*nameConstraints*», «*authorityInfoAccess*», «*subjectInfoAccess*» и «*cRLDistributionPoints*» поля «*Расширения*» сертификата, в последовательностях «*authorityInformationAccess*» и «*issuingDistributionPoint*» субполя «*crlExtensions*» («*Расширения СОС*»). Каждое

из этих расширений использует формат обобщённых имён «*GeneralName*». Единственным вариантом обобщённого имени «*GeneralName*» является *rfc822-Name*-формат с соответствующей кодировкой «*IA5String*».

Для адаптации адресов почтовой службы к текущему формату с использованием ИСНА, прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны преобразовывать почтовые адреса в *ASCII*-код.

Если часть почтового адреса, указывающая на почтовый сервер (сетевой сегмент почтового ящика, *the domain of the mailbox*), содержит ИНА, то наименование сетевого сегмента должно, в обязательном порядке, преобразовываться из ИСНА в последовательность ACE-маркеров, как это определено в §7.2.

Считается, что два адреса электронной почтовой службы совпадают, если:

1) локальные части из обоих наименования точно совпадают;

2) части почтовых адресов, указывающие на почтовые серверы, совпадают, после их сравнения в *ASCII*-кодировке в режиме игнорирования регистра написания символов.

Целесообразно, чтобы прикладные системы и ИТС преобразовывали часть почтового адреса (сам адрес содержится в расширениях сертификата или СОС), указывающую на почтовый сервер, в Юникод ещё до отображения почтового адреса на дисплее. В частности, целесообразно, чтобы прикладные системы и ИТС, придерживающиеся данного стандарта, осуществляли преобразование части адреса почтового ящика, указывающей на почтовый сервер, в соответствие с правилами, представленными в §7.2.

VIII ПРОБЛЕМЫ ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Основная часть данного стандарта посвящена конструкциям, форматам и содержанию сертификатов и СОС. Так как сертификаты и СОС подписываются с помощью ЭЦП, то нет необходимости запрашивать дополнительные услуги, предоставляемые службой обеспечения целостности. Ни сертификаты, ни СОС

не нуждаются в обеспечении конфиденциальности, и поэтому неограниченный и анонимный доступ к ним не повлекут за собой никаких последствий, связанных с обеспечением информационной безопасности (ИБ).

Тем не менее, внешние факторы, влияющие на уровень защищённости, но не рассматриваемые в данном стандарте, могут повлиять на надёжное обслуживание пользователей сертификатов.

Процедуры, реализуемые УЦ и РЦ с цель подтверждения подлинности привязки параметра подлинности владельца сертификата к его открытому ключу, в значительной степени влияют на надёжность и точность того, что должно быть указано в сертификате. Взаимодействующие стороны могут запросить для просмотра официальный отчёт о деятельности УЦ в области сертификации. Это особенно важно, когда сертификаты издаются в интересах других УЦ.

Использование одной пары ключей для подписи и в других целях категорически не допустимо. Использование отдельных пар ключей для подписи и при обеспечении ключами даёт несколько преимуществ для пользователей. Последствия, вызванные потерей или вскрытием ключа, предназначенного для процедуры формирования подписи, отличаются от потери или вскрытия ключа, предназначенного для процедуры обеспечения ключами. Использование отдельных пар ключей позволяет получить сбалансированный и приемлемый ответ. Аналогичным образом, различные периоды действия или размеры для каждой пары ключей могут быть вполне допустимы в некоторых прикладных областях. К сожалению, некоторые реальные прикладные системы (например, уровень защищённых прикладных интерфейсов, *Secure Sockets Layer* — SSL[*]) используют одну пару ключей для формирования подписи и процедур обеспечения ключами.

Защита, обеспечиваемая закрытыми ключами, является фактором риска. В малых масштабах, небрежность пользователей по защите своих закрытых ключей может предоставить нарушителю шанс для проведения атаки типа

[*] Socket — программный интерфейс доступа к динамически подключаемым к библиотекам криптографических функций, который имеет сдвоенный идентификатор, формируемый TCP/UDP-модулями с использованием пары IP-адресов и пары номеров портов транспортного уровня.

«маскарад» (под видом полномочных пользователей) или дешифровать их персональные данные. В больших масштабах, компрометация закрытого ключа УЦ, предназначенного для формирования подписи, может иметь катастрофические последствия.

Если нарушитель получил закрытый ключ УЦ «незаметно» (скрытно), то он может издавать ложные сертификаты и СОС. Даже если нарушитель не способен получить копию закрытого ключа УЦ, он может быть способен издать фальшивые сертификаты и СОС путём несанкционированного захвата и последующего противоправного использования рабочей станции УЦ или РЦ. Указанная атака может быть результатом получения нарушителем несанкционированного доступа к рабочей станции, либо локально, либо удалённо, либо может быть результатом противоправной деятельности внутреннего нарушителя (*insider*). Наличие фальшивых сертификатов и СОС подорвёт доверие к системе. Среди многих других возможных атак, существует атака, при которой нарушитель может издать ложные сертификаты в качестве легитимных, которые содержат одни и те же имена владельцев сертификатов, с целью «перевоплощения» в законных владельцев сертификатов. Такая атака могла бы привести к появлению фальшивых сертификатов УЦ, в которых имена владельцев таких фальшивых сертификатов совпадают с именами владельцев, для которых, в свою очередь, законные УЦ издали сертификаты и СОС. В последующем, такая ситуация могла бы позволить нарушителю выпустить ложные сертификаты и СОС, в которых указаны одни и те же имена владельцев сертификатов и, возможно, одни и те же последовательные номера, как и в сертификатах и СОС, изданных законными УЦ. Потеря закрытого ключа УЦ, предназначенного для формирования подписи, может также стать причиной появления многих проблем. УЦ возможно не смог бы издавать СОС или продлевать действие обычного ключа. Целесообразно, чтобы УЦ выполняли защищённое резервное копирование ключей подписи. Безопасность процедур резервного копирования ключей также является фактором риска при обеспечении защиты ключей от их компрометации.

Доступность и новизна информации об отзыве сертификатов могут сказаться на уровне доверия к содержащейся в сертификате информации. Несмотря на то, что срок действия сертификатов заканчивается естественным образом, в течение их естественного жизненного цикла могут произойти события, которые «забракуют» связку между владельцем сертификата и его открытым ключом. Если информация об отзыве сертификатов несвоевременна или недоступна, то очевидно, что надёжность привязки владельца сертификата к его открытому ключу снижается. Взаимодействующие стороны могут быть не способны обрабатывать каждое критичное расширение, которое может появиться в СОС. Целесообразно, чтобы УЦ осуществляли дополнительную обработку, когда доступность к информации об отзыве сертификатов обеспечивается только с помощью СОС, которые содержат критичные расширения, соответственно, если обработка таких расширений не предусмотрена данным стандартом. Например, если информация об отзыве сертификатов обеспечивается с помощью совместных усечённых и заполненных СОС, и при этом усечённые СОС издаются на много чаще, чем заполненные СОС, то взаимодействующие стороны, которые не могут обработать критичные расширения в рамках обработки усечённых СОС, будут не способны получить более новую информацию об отзыве сертификатов. С другой стороны, если заполненный СОС издаётся всякий раз тогда, когда издаётся усечённый СОС, то своевременная информация об отзыве сертификатов будет доступна для всех взаимодействующих сторон. Аналогично, прикладные системы и ИТС, реализующие ПрПП МС, рассмотренную в Главе VI, и которые пропускают проверку отзыва сертификатов, обеспечивают меньшую надёжность, чем те, которые данную проверку реализуют.

ПрПП МС зависит от необходимого знания открытых ключей (и иной информации) и о одном или нескольких доверенных УЦ. Решение доверять УЦ является основополагающим решением, так как оно, в конечном счёте, определяет уровень доверия к сертификату. Доверенное распространение открытых ключей надёжных УЦ (обычно в формате «само-подписанных» сертификатов)

является безопасной, автономной, но критичной процедурой, которая выходит за рамки рассматриваемых в данном стандарте проблем.

Кроме того, если ключ скомпрометирован или произошёл сбой в работе УЦ, что касается доверенных УЦ, то пользователю понадобится изменить (скорректировать) информацию, предоставляемую для выполнения ПрПП. Выбор слишком большого числа доверенных УЦ приведёт в трудностям при обработке данных о доверенных УЦ. С другой стороны, выбор только одного доверенного УЦ мог бы ограничить взаимосвязи пользователей только одним закрытым сообществом клиентов этого УЦ.

Качество функционирования самих прикладных систем и ИТС, которые обрабатывают сертификаты, также влияют на уровень предоставляемых гарантий. ПрПП МС, рассмотренный в Главе VI, полагается на целостность информации о доверенных УЦ, и в частности, на целостность открытых ключей, касающихся доверенных УЦ. Путём подмены открытых ключей, для которых нарушитель вскрыл (получил преступным путём) закрытый ключ, он мог бы вынудить пользователя получить фальшивые сертификаты.

Связка между ключом и владельцем сертификата не может быть более стойкой, чем стойкость, обеспечиваемая используемыми криптомодулем и алгоритмами формирования подписи. Длина коротких ключей или слабые алгоритмы вычисления хэш-функции будут ограничивать практичность и эффективность сертификата. УЦ следует следить за достижениями в криптологии, так как они могут применять более стойкие криптоалгоритмы и криптосистемы. Кроме того, целесообразно, чтобы УЦ отказывались издавать сертификаты для УЦ или конечных пользователей, которые формируют не стойкие (или не надёжные) подписи.

Не корректное исполнение прикладной системой или ИТС правил сравнения наименований (в соответствие с Рекомендацией ITU-T X.509) может привести к образованию недопустимых МС или наоборот, к удалению приемлемых МС. Рекомендации ITU-T серии X.500 устанавливают правила сравнения уникальных имён, которые требуют сравнение последовательностей сим-

волов без учёта регистра их написания, алфавита, субпоследовательностей, состоящих из нескольких пробелов или начальных и завершающих пробелов. Данный стандарт «ослабляет» эти требования, требу, как минимум обработку (сравнение) бинарных последовательностей.

УЦ обязаны кодировать уникальное имя, расположенное в поле «*Subject*» сертификата УЦ, аналогично кодированию уникального имени, расположенного в поле «*Subject*» сертификата, выпущенного эти УЦ. Если УЦ используют различные кодировки, то прикладные системы и ИТС могут не распознать имени в последовательности имён МС, в который включён данный сертификат. И как следствие, приемлемые МС могут быть удалены.

Кроме того, ограничения (субполе «*nameConstraints*» в поле «*Расширения*» сертификата) для уникальных имён должны быть закодированы, в обязательном порядке, точно так же, как закодированы поле «*Subject*» и субполе «*subjectAltName*» в поле «Расширения» сертификата. Если же нет, то субполя «*nameConstraints*», определяемые как последовательности «*excludedSubtrees*», не будут совпадать, а недопустимые МС станут приемлемыми, и субполя «*nameConstraints*», определяемые как последовательности «*permittedSubtrees*», не будут совпадать, а допустимые МС будут исключены. Чтобы избежать применения неприемлемых маршрутов, целесообразно, чтобы УЦ, там где это возможно, устанавливали ограничения для уникальных имён как последовательности «*permittedSubtrees*».

В общем, использование субполе «*nameConstraints*» в поле «*Расширения*» сертификата для ограничения одного формата наименования (например, DNS-имена) не предполагает ограничение применения других форматов имён (например, адреса электронной почтовой службы).

Несмотря на то, что сертификаты в соответствие с Рекомендацией ITU-T X.509 включают точно выраженные однозначные имена, существует риск того, что не связанные друг с другом УЦ будут издавать сертификаты и/или СОС с одним и тем же именем издателя. В качестве средств снижения остроты проблемы и обеспечения безопасности, связанных с дублированием имён издате-

лей, целесообразно, чтобы имена УЦ и издателя СОС формировались бы таким образом, чтобы снижалась вероятность дублирования имён. Целесообразно, чтобы прикладные системы и ИТС обращали внимание на возможные расширения нескольких не связанных между собой УЦ и издателей СОС, содержащих одно и то же имя. Как минимум, прикладные системы и ИТС, реализующие ПрПП СОС, обязаны гарантировать, что МС сертификата и МС издателя СОС, используемые для подтверждения подлинности сертификата, завершаются в одном и том же ДЗУЦ.

Несмотря на то, что локальная часть адреса электронной почтовой службы зависит от регистра написания символов (RFC-2821), значения атрибута «*emailAddress*» не зависят от такого регистра (RFC-2985). В этой связи, существует риск того, что два различных почтовых адреса будут считаться как один и тот же адрес, когда используется правило сравнения, предназначенное для атрибута «*emailAddress*», если почтовый сервер учитывает в процедуре сравнения локальных частей адресов почтового ящика регистр написания символов. Целесообразно, чтобы прикладные системы и ИТС не включали почтовые адреса в атрибут «*emailAddress*», если почтовый сервер, который обрабатывает почтовый адрес, рассматривает локальную часть почтовых адресов, как зависимую от регистра написания символов.

Целесообразно, чтобы прикладные системы и ИТС были осведомлены относительно рисков, возникающих с том случае, когда субполя «*cRLDistributionPoints*» и «*authorityInfoAccess*» поля «*Расширения*» фальсифицированных сертификатов или последовательности «*authorityInformationAccess*» и «*issuingDistributionPoint*» субполя «*crlExtensions*» фальсифицированных СОС содержат указатели на вредоносный код. Целесообразно, чтобы прикладные системы и ИТС всегда предпринимали шаги по проверке подлинности полученных данных с целью обеспечения подтверждения того, что данные сформированы должным образом.

Когда сертификаты включают субполе «*cRLDistributionPoints*», содержащее URI-идентификатор HTTPS-схемы адресации или аналогичной схемы, то-

гда может возникнуть циклическая взаимозависимость («зацикливание»). Взаимодействующая сторона будет вынуждена проводить дополнительную ПрПП МС с целью получения необходимого СОС для завершения первичной ПрПП МС! Условия цикличности могут также возникнуть, если и субполя «*authorityInfoAccess*» и «*subjectInfoAccess*» в поле «*Расширения*» сертификата содержат URI-идентификатор HTTPS-схемы адресации или аналогичной схемы. В худшем случае, такая ситуация может спровоцировать неразрешимую взаимосвязь (т.е. «нельзя будет разорвать порочный круг»).

Целесообразно, чтобы УЦ не включали URI-идентификаторы в расширения сертификатов и СОС, если такие идентификаторы указывают на LDAPS- и HTTPS-схемы адресации или аналогичные схемы. УЦ, которые включают URI-идентификаторы HTTPS-схемы адресации в указанные выше расширения, обязаны гарантировать, что сертификат сервера может провести ПрПП без использования информации, на которую указывает URI-идентификатор. Взаимодействующие стороны, реализующие ПрПП сертификата сервера, когда они получают информацию с помощью URI-идентификатора HTTPS-схемы адресации, содержащегося в субполях «*cRLDistributionPoints*», «*authorityInfoAccess*» и «*subjectInfoAccess*» в поле «*Расширения*» сертификата, обязаны быть готовыми к возникновению ситуации, которая может привести к бесконечной рекурсии.

Само-изданные сертификаты являются одним из автоматизированных способов, который позволяет УЦ указывать на изменения в их функциональных характеристиках. Соответственно, само-изданные сертификаты могут использоваться в качестве инструмента для «постепенного перехода» от одной нескомпрометированной пары ключей УЦ к другой. Более детальные процедуры обновления ключей УЦ представлены в стандарте RFC-4210, которые позволяют УЦ защитить свой новый открытый ключ, используя свой предшествующий закрытый ключ, и наоборот, используя два само-изданных сертификата. Прикладные системы и ИТС, обслуживающие клиентов и придерживающиеся данного стандарта, будут обрабатывать само-изданные сертификаты и устанавливать, могут ли быть доверенными (надёжными) сертификаты, которые были

изданы с использованием нового ключа. Само-изданные сертификаты могут применяться для указания и других изменений в функциональных характеристиках УЦ, например, дополнения к совокупности политик УЦ, с использованием аналогичных процедур.

Некоторые «старые» прикладные системы и ИТС обрабатывают имена, которые закодированы с помощью набора символов, установленного стандартом ISO 8859-1 («*Latin1String*», латинский алфавит), но помечают эти имена, как если бы они имели *TeletexString*-кодировку. Алфавит *TeletexString*-кодировки имеет большее число символов по сравнению с алфавитом, установленным стандартом ISO 8859-1. Более того, в алфавите *TeletexString*-кодировки некоторые символы кодируются по другому, по сравнению со стандартом ISO 8859-1. Правила сравнения имён, представленные в §7.1, предусматривают, что алфавит *TeletexString*-кодировки соответствует правилам кодирования ASN.1-стандарта. При сравнении имён, закодированных с помощью *Latin1String*-алфавита, возможны ошибочные, и положительные, и отрицательные решения.

Если последовательности символов отображаются из внутреннего кода в код визуального отображения, то иногда две различные последовательности могут иметь одинаковое или подобное визуальное отображение. Такое может произойти по многим причинам, включая представление символа с помощью побитового способа отображения (например, «е + '», эквивалентное U+00E9-символу (0000 0000 1110 1001), Корейские составные символы и гласные над группами согласных в некоторых языках). В результате возникновения такой ситуации пользователи, осуществляющие визуальное сравнение двух различных наименований могут подумать, что они одинаковые, а самом деле — нет. Кроме того, пользователи могут ошибаться принимая одну последовательность символов за другую. Издатели сертификатов и взаимодействующие стороны должны быть осведомлены о такой проблеме.

Однонаправленные хэш-функции, как правило, используются при формировании значение идентификаторов ключей (*Authority Key Identifier* и *Subject*

Key Identifier), например, как определено в §4.1.1 и §4.1.2. Тем не менее, в данном стандарте не требуется указывать ни одно из свойств таких функций.

Приложение А

В данном приложении описываются информационные объекты, используемые компонентами Интернет/PKI-инфраструктуры, с применением синтаксиса подобного *ASN.1*-формату. Такой синтаксис представляет собой гибрид синтаксисов *ASN.1*-форматов, соответствующих стандартам различных годов, а именно 1988 и 1993 г.г. Синтаксис 1988 г. дополняется синтаксисом 1993 г. за счёт введения универсальных форматов данных («*UNIVERSAL*»), а именно *UniversalString*-, *BMPString*- и *UTF8String*-кодировок.

Синтаксис *ASN.1*-формата не допускает включение форматов «*statements*» (операторы) в *ASN.1*-модуль, а стандарт 1993 г. не допускает использование в модулях на основе синтаксиса 1988 г. новых универсальных форматов данных («*UNIVERSAL*»). В этой связи, представленный далее модуль не соответствует ни одной из версий *ASN.1*-стандарта.

Это приложение (*ASN.1*-модули) может быть преобразовано в *ASN.1*-стандарт 1988 г. путём замены описаний универсальных форматов данных («*UNIVERSAL*») на обобщённое наименование «*ANY*» из стандарта 1988 г.

§A.1 Точно размеченный ASN.1-модуль, синтаксис 1988 г.

```
PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

-- IMPORTS NONE --

-- форматы данных «UNIVERSAL», установленные стандартами ASN.1 1993 и 1998 г.г.
-- и востребованные данным стандартом

UniversalString ::= [UNIVERSAL 28] IMPLICIT OCTET STRING
      -- UniversalString-формат представлен в стандарте ASN.1:1993
```

```
BMPString ::=           [UNIVERSAL 30] IMPLICIT OCTET STRING
    -- BMPString-формат, являющийся подвидом UniversalString-формата, и модели
    -- Основной многоязычный символьный алфавит,
    -- в соответствие со стандартом ISO/IEC 10646

UTF8String ::=          [UNIVERSAL 12] IMPLICIT OCTET STRING
    -- содержание этого формата соответствует стандарту RFC 3629.

-- специальные OID в интересах Интернет/PKI-инфраструктуры

id-pkix                 OBJECT IDENTIFIER  ::=
      { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) }

-- ветви дерева иерархии Интернет/PKI-инфраструктуры

id-pe                   OBJECT IDENTIFIER ::= { id-pkix 1 }
      -- ветвь частных расширений для дерева иерархии Интернет/PKI-инфраструктуры
id-qt                   OBJECT IDENTIFIER ::= { id-pkix 2 }
      -- ветвь форматов определителя политик
id-kp                   OBJECT IDENTIFIER ::= { id-pkix 3 }
      -- ветвь OID расширения области действия ключа
id-ad                   OBJECT IDENTIFIER ::= { id-pkix 48 }
      -- ветвь определителей доступа

-- идентификаторы определителей политик для Интернет/PKI-инфраструктуры

id-qt-cps               OBJECT IDENTIFIER ::=  { id-qt 1 }
    -- OID CPS-указателя
id-qt-unotice           OBJECT IDENTIFIER ::=  { id-qt 2 }
    -- OID определителя извещения пользователя

-- описания определителя доступа

id-ad-ocsp              OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers         OBJECT IDENTIFIER ::= { id-ad 2 }
id-ad-timeStamping      OBJECT IDENTIFIER ::= { id-ad 3 }
id-ad-caRepository      OBJECT IDENTIFIER ::= { id-ad 5 }

-- типы (форматы) данных в атрибутах

Attribute           ::= SEQUENCE {
            type                    AttributeType,
            values                  SET OF AttributeValue }
        -- по крайней мере требуется одно значение

AttributeType           ::= OBJECT IDENTIFIER

AttributeValue          ::= ANY -- DEFINED BY AttributeType

AttributeTypeAndValue   ::= SEQUENCE {
            type                    AttributeType,
            value                   AttributeValue }

-- Предлагаемые атрибуты именования: Описание следующей совокупности
-- информационных объектов может быть дополнено с целью удовлетворения
-- локальных требований. Следует отметить, что удаление элементов
-- этой совокупности может нарушить функциональную совместимость
-- с прикладными системами и ИТС, придерживающимися данного стандарта.
-- Состоят из двух полей: за полем «AttributeType» (тип атрибута) следует
-- определение формата соответствующего поля «AttributeValue» (значение атрибута)
-- Ветвь стандартных атрибутов наименований
```

```
id-at                       OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }

-- атрибуты наименований формата «X520name»

id-at-name                  AttributeType ::= { id-at 41 }
id-at-surname               AttributeType ::= { id-at  4 }
id-at-givenName             AttributeType ::= { id-at 42 }
id-at-initials              AttributeType ::= { id-at 43 }
id-at-generationQualifier   AttributeType ::= { id-at 44 }

-- Атрибуты наименований формата «X520name»:
--   X520name ::= DirectoryString (SIZE (1..ub-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520name ::=                CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-name)),
    printableString         PrintableString (SIZE (1..ub-name)),
    universalString         UniversalString (SIZE (1..ub-name)),
    utf8String              UTF8String      (SIZE (1..ub-name)),
    bmpString               BMPString       (SIZE (1..ub-name)) }

-- атрибуты наименований формата «X520CommonName»

id-at-commonName            AttributeType ::= { id-at 3 }

-- Атрибуты наименований формата «X520CommonName»:
--   X520CommonName ::= DirectoryName (SIZE (1..ub-common-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520CommonName ::=          CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-common-name)),
    printableString         PrintableString (SIZE (1..ub-common-name)),
    universalString         UniversalString (SIZE (1..ub-common-name)),
    utf8String              UTF8String      (SIZE (1..ub-common-name)),
    bmpString               BMPString       (SIZE (1..ub-common-name)) }

-- атрибуты наименований формата «X520LocalityName»

id-at-localityName          AttributeType ::= { id-at 7 }

-- Атрибуты наименований формата «X520LocalityName»:
--   X520LocalityName ::= DirectoryName (SIZE (1..ub-locality-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520LocalityName ::=        CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-locality-name)),
    printableString         PrintableString (SIZE (1..ub-locality-name)),
    universalString         UniversalString (SIZE (1..ub-locality-name)),
    utf8String              UTF8String      (SIZE (1..ub-locality-name)),
    bmpString               BMPString       (SIZE (1..ub-locality-name)) }

-- атрибуты наименований формата «X520StateOrProvinceName»

id-at-stateOrProvinceName   AttributeType ::= { id-at 8 }

-- Атрибуты наименований формата «X520StateOrProvinceName»:
--   X520StateOrProvinceName ::= DirectoryName (SIZE (1..ub-state-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520StateOrProvinceName ::= CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-state-name)),
    printableString         PrintableString (SIZE (1..ub-state-name)),
```

```
    universalString             UniversalString (SIZE (1..ub-state-name)),
    utf8String                  UTF8String      (SIZE (1..ub-state-name)),
    bmpString                   BMPString       (SIZE (1..ub-state-name)) }

-- атрибуты наименований формата «X520OrganizationName»

id-at-organizationName          AttributeType ::= { id-at 10 }

-- Атрибуты наименований формата «X520OrganizationName»:
--   X520OrganizationName ::=
--        DirectoryName (SIZE (1..ub-organization-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520OrganizationName ::=    CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-organization-name)),
    printableString         PrintableString (SIZE (1..ub-organization-name)),
    universalString         UniversalString (SIZE (1..ub-organization-name)),
    utf8String              UTF8String      (SIZE (1..ub-organization-name)),
    bmpString               BMPString       (SIZE (1..ub-organization-name))  }

-- атрибуты наименований формата «X520OrganizationalUnitName»

id-at-organizationalUnitName    AttributeType ::= { id-at 11 }

-- Атрибуты наименований формата «X520OrganizationalUnitName»:
--   X520OrganizationalUnitName ::=
--        DirectoryName (SIZE (1..ub-organizational-unit-name))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520OrganizationalUnitName ::=  CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-organizational-unit-name)),
    printableString         PrintableString (SIZE (1..ub-organizational-unit-name)),
    universalString         UniversalString (SIZE (1..ub-organizational-unit-name)),
    utf8String              UTF8String      (SIZE (1..ub-organizational-unit-name)),
    bmpString               BMPString       (SIZE (1..ub-organizational-unit-name)) }

-- атрибуты наименований формата «X520Title»

id-at-title                     AttributeType ::= { id-at 12 }

-- Атрибуты наименований формата «X520Title»:
--   X520Title ::= DirectoryName (SIZE (1..ub-title))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520Title ::=               CHOICE {
    teletexString           TeletexString   (SIZE (1..ub-title)),
    printableString         PrintableString (SIZE (1..ub-title)),
    universalString         UniversalString (SIZE (1..ub-title)),
    utf8String              UTF8String      (SIZE (1..ub-title)),
    bmpString               BMPString       (SIZE (1..ub-title)) }

-- атрибуты наименований формата «X520dnQualifier»

id-at-dnQualifier               AttributeType ::= { id-at 46 }

X520dnQualifier ::=             PrintableString

-- атрибуты наименований формата «X520countryName» (digraph from IS 3166)

id-at-countryName               AttributeType ::= { id-at 6 }

X520countryName ::=             PrintableString  (SIZE (2))
```

```
-- атрибуты наименований формата «X520SerialNumber»

id-at-serialNumber                AttributeType ::= { id-at 5 }

X520SerialNumber ::=              PrintableString (SIZE (1..ub-serial-number))

-- атрибуты наименований формата «X520Pseudonym»

id-at-pseudonym                   AttributeType ::= { id-at 65 }

-- Атрибуты наименований формата «X520Pseudonym»:
--   X520Pseudonym ::= DirectoryName (SIZE (1..ub-pseudonym))
--
-- Расширение с целью исключения записей в параметрическом формате:
X520Pseudonym ::=                 CHOICE {
    teletexString                 TeletexString   (SIZE (1..ub-pseudonym)),
    printableString               PrintableString (SIZE (1..ub-pseudonym)),
    universalString               UniversalString (SIZE (1..ub-pseudonym)),
    utf8String                    UTF8String      (SIZE (1..ub-pseudonym)),
    bmpString                     BMPString       (SIZE (1..ub-pseudonym)) }

-- атрибуты наименований формата «DomainComponent» (from RFC 4519)

id-domainComponent                AttributeType ::= { 0 9 2342 19200300 100 1 25 }

DomainComponent ::=               IA5String

-- унаследованные атрибуты

pkcs-9                        OBJECT IDENTIFIER ::=
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 9 }

id-emailAddress                   AttributeType ::= { pkcs-9 1 }

EmailAddress ::=                  IA5String       (SIZE (1..ub-emailaddress-length))

-- форматы данных наименований --

Name ::=                          CHOICE { -- в настоящее время возможен только один --
                                  rdnSequence  RDNSequence }

RDNSequence ::=                   SEQUENCE OF RelativeDistinguishedName

DistinguishedName ::=             RDNSequence

RelativeDistinguishedName ::=     SET SIZE (1..MAX) OF AttributeTypeAndValue

-- формат последовательности символов для Службы единого каталога --

DirectoryString ::=               CHOICE {
    teletexString                 TeletexString   (SIZE (1..MAX)),
    printableString               PrintableString (SIZE (1..MAX)),
    universalString               UniversalString (SIZE (1..MAX)),
    utf8String                    UTF8String      (SIZE (1..MAX)),
    bmpString                     BMPString       (SIZE (1..MAX)) }

-- конкретные структуры сертификата и СОС начинаются здесь

Certificate  ::=              SEQUENCE  {
            tbsCertificate        TBSCertificate,
            signatureAlgorithm    AlgorithmIdentifier,
```

```
        signature               BIT STRING  }

TBSCertificate  ::=     SEQUENCE  {
    version             [0]     Version DEFAULT v1,
    serialNumber                CertificateSerialNumber,
    signature                   AlgorithmIdentifier,
    issuer                      Name,
    validity                    Validity,
    subject                     Name,
    subjectPublicKeyInfo        SubjectPublicKeyInfo,
    issuerUniqueID      [1]     IMPLICIT UniqueIdentifier OPTIONAL,
                -- если представлен, то должен быть второй или третьей версии
    subjectUniqueID     [2]     IMPLICIT UniqueIdentifier OPTIONAL,
                -- если представлен, то должен быть второй или третьей версии
    extensions          [3]     Extensions OPTIONAL
                -- если представлен, то должен быть третьей версии --  }

Version  ::=            INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=    INTEGER

Validity ::=            SEQUENCE {
    notBefore                   Time,
    notAfter                    Time  }

Time ::=                CHOICE {
    utcTime                     UTCTime,
    generalTime                 GeneralizedTime }

UniqueIdentifier  ::=   BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
    algorithm                   AlgorithmIdentifier,
    subjectPublicKey            BIT STRING  }

Extensions  ::=         SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=          SEQUENCE  {
    extnID                      OBJECT IDENTIFIER,
    critical                    BOOLEAN DEFAULT FALSE,
    extnValue                   OCTET STRING
            -- содержит ASN.1-модуль в DER-кодировке, соответствующей расширенному
            -- формату, определяемому идентификатором расширения «extnID»
    }

-- Структуры СОС

CertificateList  ::=    SEQUENCE  {
    tbsCertList                 TBSCertList,
    signatureAlgorithm          AlgorithmIdentifier,
    signature                   BIT STRING  }

TBSCertList  ::=        SEQUENCE  {
    version                     Version OPTIONAL,
                        -- если представлена, то должна быть вторая версия
    signature                   AlgorithmIdentifier,
    issuer                      Name,
    thisUpdate                  Time,
    nextUpdate                  Time OPTIONAL,
    revokedCertificates         SEQUENCE OF SEQUENCE  {
            userCertificate             CertificateSerialNumber,
            revocationDate              Time,
```

```
            crlEntryExtensions              Extensions OPTIONAL
                        -- если представлена, то должна быть вторая версия
                    }  OPTIONAL,
    crlExtensions       [0] Extensions OPTIONAL }
                        -- если представлена, то должна быть вторая версия

-- поля Version, Time, CertificateSerialNumber и Extensions были определены ранее,
-- в структуре сертификата

AlgorithmIdentifier  ::=    SEQUENCE  {
    Algorithm                       OBJECT IDENTIFIER,
    Parameters                      ANY DEFINED BY algorithm OPTIONAL  }
                        -- содержит значение формата, который зарегистрирован
                        -- для совместного использования со значением
                        -- идентификатора алгоритма

-- синтаксис X.400-адресов начинается здесь

ORAddress ::=               SEQUENCE {
    built-in-standard-attributes    BuiltInStandardAttributes, built-in-domain-defined-attributes
                                    BuiltInDomainDefinedAttributes OPTIONAL,
  -- см. также атрибуты для teletext, определяемые сетевым сегментом/областью
    extension-attributes            ExtensionAttributes OPTIONAL }

-- стандартные встраиваемые атрибуты

BuiltInStandardAttributes ::=   SEQUENCE {
    country-name                        CountryName OPTIONAL,
    administration-domain-name          AdministrationDomainName OPTIONAL,
    network-address             [0]     IMPLICIT NetworkAddress OPTIONAL,
    -- см. также расширенный сетевой адрес
    terminal-identifier         [1]     IMPLICIT TerminalIdentifier OPTIONAL,
    private-domain-name         [2]     PrivateDomainName OPTIONAL,
    organization-name           [3]     IMPLICIT OrganizationName OPTIONAL,
    -- см. также наименование организации в teletex-формате
    numeric-user-identifier     [4]     IMPLICIT NumericUserIdentifier OPTIONAL,
    personal-name               [5]     IMPLICIT PersonalName OPTIONAL,
    -- см. также персональное имя в teletex-формате
    organizational-unit-names   [6]     IMPLICIT OrganizationalUnitNames
                                        OPTIONAL }
    -- см. также наименования подразделений организации в teletex-формате

CountryName ::=             [APPLICATION 1]     CHOICE {
    x121-dcc-code               NumericString
                                (SIZE (ub-country-name-numeric-length)),
    iso-3166-alpha2-code        PrintableString
                                (SIZE (ub-country-name-alpha-length)) }

AdministrationDomainName ::=    [APPLICATION 2]     CHOICE {
    Numeric                     NumericString (SIZE (0..ub-domain-name-length)),
    Printable                   PrintableString (SIZE (0..ub-domain-name-length)) }

NetworkAddress ::=          X121Address  -- см. также расширенный сетевой адрес

X121Address ::=             NumericString (SIZE (1..ub-x121-address-length))

TerminalIdentifier ::=      PrintableString (SIZE (1..ub-terminal-id-length))

PrivateDomainName ::=               CHOICE {
    Numeric                     NumericString (SIZE (1..ub-domain-name-length)),
    Printable                   PrintableString (SIZE (1..ub-domain-name-length)) }
```

```
OrganizationName ::=            PrintableString (SIZE (1..ub-organization-name-length))
  -- см. также наименование организации в teletex-формате

NumericUserIdentifier ::=       NumericString (SIZE (1..ub-numeric-user-id-length))

PersonalName ::=                SET {
    Surname                     [0]     IMPLICIT PrintableString
                                        (SIZE (1..ub-surname-length)),
    given-name                  [1]     IMPLICIT PrintableString
                                        (SIZE (1..ub-given-name-length)) OPTIONAL,
    Initials                    [2]     IMPLICIT PrintableString
                                        (SIZE (1..ub-initials-length)) OPTIONAL,
    generation-qualifier        [3]     IMPLICIT PrintableString
                                        (SIZE (1..ub-generation-qualifier-length))
                                        OPTIONAL }
  -- см. также персональное имя в teletex-формате

OrganizationalUnitNames ::=     SEQUENCE SIZE (1..ub-organizational-units)
                                OF OrganizationalUnitName
  -- см. также наименования подразделений организации в teletex-формате

OrganizationalUnitName ::=      PrintableString (SIZE
                                (1..ub-organizational-unit-name-length))

-- встраиваемые атрибуты, определяемые сетевыми сегментами/областями

BuiltInDomainDefinedAttributes ::= SEQUENCE SIZE
                                        (1..ub-domain-defined-attributes) OF
                                        BuiltInDomainDefinedAttribute

BuiltInDomainDefinedAttribute ::=       SEQUENCE {
    type PrintableString (SIZE (1..ub-domain-defined-attribute-type-length)),
    value PrintableString (SIZE (1..ub-domain-defined-attribute-value-length)) }

-- атрибуты расширений

ExtensionAttributes ::=         SET SIZE (1..ub-extension-attributes) OF
                                ExtensionAttribute

ExtensionAttribute ::=          SEQUENCE {
    extension-attribute-type [0]    IMPLICIT INTEGER
                                        (0..ub-extension-attributes),
    extension-attribute-value   [1]     ANY DEFINED BY extension-attribute-type }

-- форматы расширений и значения атрибутов

common-name                     INTEGER ::= 1

CommonName ::=          PrintableString (SIZE (1..ub-common-name-length))

teletex-common-name             INTEGER ::= 2

TeletexCommonName ::=           TeletexString (SIZE (1..ub-common-name-length))

teletex-organization-name       INTEGER ::= 3

TeletexOrganizationName ::=
                                TeletexString (SIZE (1..ub-organization-name-length))

teletex-personal-name           INTEGER ::= 4

TeletexPersonalName ::=         SET {
```

```
    surname                         [0]     IMPLICIT TeletexString
                                            (SIZE (1..ub-surname-length)),
    given-name                      [1]     IMPLICIT TeletexString
                                            (SIZE (1..ub-given-name-length)) OPTIONAL,
    initials                        [2]     IMPLICIT TeletexString
                                            (SIZE (1..ub-initials-length)) OPTIONAL,
    generation-qualifier            [3]     IMPLICIT TeletexString
                                            (SIZE (1..ub-generation-qualifier-length))
                                            OPTIONAL }

teletex-organizational-unit-names   INTEGER ::= 5

TeletexOrganizationalUnitNames ::=      SEQUENCE SIZE
                    (1..ub-organizational-units) OF TeletexOrganizationalUnitName

TeletexOrganizationalUnitName ::=       TeletexString
                    (SIZE (1..ub-organizational-unit-name-length))

pds-name                            INTEGER ::= 7

PDSName ::=                         PrintableString (SIZE (1..ub-pds-name-length))

physical-delivery-country-name      INTEGER ::= 8

PhysicalDeliveryCountryName ::=         CHOICE {
    x121-dcc-code NumericString (SIZE (ub-country-name-numeric-length)),
    iso-3166-alpha2-code PrintableString (SIZE (ub-country-name-alpha-length)) }

postal-code                         INTEGER ::= 9

PostalCode ::=                              CHOICE {
    numeric-code                    NumericString (SIZE (1..ub-postal-code-length)),
    printable-code                  PrintableString (SIZE (1..ub-postal-code-length)) }

physical-delivery-office-name       INTEGER ::= 10

PhysicalDeliveryOfficeName ::=          PDSParameter

physical-delivery-office-number     INTEGER ::= 11

PhysicalDeliveryOfficeNumber ::=        PDSParameter

extension-OR-address-components         INTEGER ::= 12

ExtensionORAddressComponents ::=    PDSParameter

physical-delivery-personal-name     INTEGER ::= 13

PhysicalDeliveryPersonalName ::= PDSParameter

physical-delivery-organization-name     INTEGER ::= 14

PhysicalDeliveryOrganizationName ::=    PDSParameter

extension-physical-delivery-address-components          INTEGER ::= 15

ExtensionPhysicalDeliveryAddressComponents ::= PDSParameter

unformatted-postal-address          INTEGER ::= 16

UnformattedPostalAddress ::=            SET {
    printable-address               SEQUENCE SIZE (1..ub-pds-physical-address-lines)
```

```
                    OF PrintableString (SIZE (1..ub-pds-parameter-length)) OPTIONAL,
    teletex-string                  TeletexString
                                    (SIZE (1..ub-unformatted-address-length)) OPTIONAL }

street-address                      INTEGER ::= 17

StreetAddress ::=                       PDSParameter

post-office-box-address             INTEGER ::= 18

PostOfficeBoxAddress ::=                PDSParameter

poste-restante-address              INTEGER ::= 19

PosteRestanteAddress ::=            PDSParameter

unique-postal-name                  INTEGER ::= 20

UniquePostalName ::=                    PDSParameter

local-postal-attributes             INTEGER ::= 21

LocalPostalAttributes ::=               PDSParameter

PDSParameter ::=                        SET {
    printable-string                    PrintableString
                                        (SIZE(1..ub-pds-parameter-length)) OPTIONAL,
    teletex-string                      TeletexString
                                        (SIZE(1..ub-pds-parameter-length)) OPTIONAL }

extended-network-address INTEGER ::= 22

ExtendedNetworkAddress ::=              CHOICE {
    e163-4-address                  SEQUENCE {
    number              [0]         IMPLICIT NumericString
                                        (SIZE (1..ub-e163-4-number-length)),
    sub-address                     [1]     IMPLICIT NumericString
                                        (SIZE (1..ub-e163-4-sub-address-length))
                                        OPTIONAL },
    psap-address                    [0]     IMPLICIT PresentationAddress }

PresentationAddress ::=                 SEQUENCE {
    pSelector                       [0]     EXPLICIT OCTET STRING OPTIONAL,
    sSelector                       [1]     EXPLICIT OCTET STRING OPTIONAL,
    tSelector                       [2]     EXPLICIT OCTET STRING OPTIONAL,
    nAddresses                      [3]     EXPLICIT SET SIZE (1..MAX)
                                            OF OCTET STRING }

terminal-type                       INTEGER ::= 23

TerminalType ::=                    INTEGER {
    telex                           (3),
    teletext                        (4),
    g3-facsimile                    (5),
    g4-facsimile                    (6),
    ia5-terminal                    (7),
    videotext                       (8) } (0..ub-integer-options)

-- встраиваемые атрибуты, определяемые сетевыми сегментами/областями

teletex-domain-defined-attributes       INTEGER ::= 6
```

```
TeletexDomainDefinedAttributes ::=          SEQUENCE SIZE
            (1..ub-domain-defined-attributes) OF TeletexDomainDefinedAttribute

TeletexDomainDefinedAttribute ::= SEQUENCE {
    type TeletexString              (SIZE (1..ub-domain-defined-attribute-type-length)),
    value TeletexString             (SIZE (1..ub-domain-defined-attribute-value-length)) }

--  описание верхних границ в Приложении B Рекомендации ITU-T X.411
--  (Reference Definition of MTS Parameter Upper Bounds, Эталонное описание верхних
--  границ параметров в системах доставки сообщений) обязательно должно
--  рассматриваться как эталонное
--
-- Верхние границы
ub-name                                 INTEGER ::= 32768
ub-common-name                          INTEGER ::= 64
ub-locality-name                        INTEGER ::= 128
ub-state-name                           INTEGER ::= 128
ub-organization-name                    INTEGER ::= 64
ub-organizational-unit-name             INTEGER ::= 64
ub-title                                INTEGER ::= 64
ub-serial-number                        INTEGER ::= 64
ub-match                                INTEGER ::= 128
ub-emailaddress-length                  INTEGER ::= 255
ub-common-name-length                   INTEGER ::= 64
ub-country-name-alpha-length            INTEGER ::= 2
ub-country-name-numeric-length          INTEGER ::= 3
ub-domain-defined-attributes            INTEGER ::= 4
ub-domain-defined-attribute-type-length INTEGER ::= 8
ub-domain-defined-attribute-value-length INTEGER ::= 128
ub-domain-name-length                   INTEGER ::= 16
ub-extension-attributes                 INTEGER ::= 256
ub-e163-4-number-length                 INTEGER ::= 15
ub-e163-4-sub-address-length            INTEGER ::= 40
ub-generation-qualifier-length          INTEGER ::= 3
ub-given-name-length                    INTEGER ::= 16
ub-initials-length                      INTEGER ::= 5
ub-integer-options                      INTEGER ::= 256
ub-numeric-user-id-length               INTEGER ::= 32
ub-organization-name-length             INTEGER ::= 64
ub-organizational-unit-name-length INTEGER ::= 32
ub-organizational-units                 INTEGER ::= 4
ub-pds-name-length                      INTEGER ::= 16
ub-pds-parameter-length                 INTEGER ::= 30
ub-pds-physical-address-lines           INTEGER ::= 6
ub-postal-code-length                   INTEGER ::= 16
ub-pseudonym                            INTEGER ::= 128
ub-surname-length               INTEGER ::= 40
ub-terminal-id-length                   INTEGER ::= 24
ub-unformatted-address-length           INTEGER ::= 180
ub-x121-address-length                  INTEGER ::= 16

-- Примечание – верхние границы форматов последовательностей символов,
-- например TeletexString, измеряются в символах. За исключением
-- последовательностей PrintableString или IA5String, для хранения
-- значения потребуется значительно большее количество октетов.
-- Целесообразно, чтобы для последовательности TeletexString было
-- зарезервировано как минимум 16 октетов или в два раза больше в качестве
-- верхней границы. Целесообразно, чтобы для последовательности UTF8String или
-- UniversalString было зарезервировано, как минимум, четырёх кратное превышение
-- верхней границы.

END
```

§A.2 Размеченный по содержанию ASN.1-модуль, синтаксис 1988 г.

```
PKIX1Implicit88     { iso(1) identified-organization(3) dod(6) internet(1)
                      security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit(19) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS
    id-pe, id-kp, id-qt-unotice, id-qt-cps,
    -- удаляем следующую строку, если обрабатываются «новые» форматы --
    BMPString, UTF8String,  -- окончание «новые» форматов --
    ORAddress, Name, RelativeDistinguishedName,
    CertificateSerialNumber, Attribute, DirectoryString
    FROM PKIX1Explicit88{ iso(1) identified-organization(3)
                          dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                          id-mod(0) id-pkix1-explicit(18) };

-- ISO-ветвь дерева иерархии для стандартных расширений сертификата и СОС

id-ce                               OBJECT IDENTIFIER  ::=  {joint-iso-ccitt(2) ds(5) 29}

-- идентификатор и синтаксис ключа УЦ

id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

AuthorityKeyIdentifier ::=          SEQUENCE {
    keyIdentifier                   [0]     KeyIdentifier           OPTIONAL,
    authorityCertIssuer             [1]     GeneralNames            OPTIONAL,
    authorityCertSerialNumber       [2]     CertificateSerialNumber  OPTIONAL }
   -- издатель и последовательный номер сертификата УЦ, либо
   -- должны обязательно присутствовать вместе, либо должны обязательно
   -- одновременно отсутствовать

KeyIdentifier ::=                   OCTET STRING

-- идентификатор и синтаксис ключа владельца сертификата

id-ce-subjectKeyIdentifier          OBJECT IDENTIFIER ::=  { id-ce 14 }

SubjectKeyIdentifier ::=            KeyIdentifier

-- идентификатор и синтаксис расширения «область применения ключа»

id-ce-keyUsage                      OBJECT IDENTIFIER ::=  { id-ce 15 }

KeyUsage ::=                        BIT STRING {
    digitalSignature (0),
    nonRepudiation(1),      -- в последних изданиях Рекомендации ITU-T X.509,
                            -- этот бит переименован на «contentCommitment»
    keyEncipherment     (2),
    dataEncipherment    (3),
    keyAgreement        (4),
    keyCertSign         (5),
    cRLSign             (6),
    encipherOnly        (7),
    decipherOnly        (8) }
```

```
-- идентификатор и синтаксис срока действия закрытого ключа

id-ce-privateKeyUsagePeriod     OBJECT IDENTIFIER ::=  { id-ce 16 }

PrivateKeyUsagePeriod ::=               SEQUENCE {
    notBefore                   [0]     GeneralizedTime     OPTIONAL,
    notAfter                [1]     GeneralizedTime     OPTIONAL }
    -- обязательно должно присутствовать, либо время «notBefore»,
    -- либо время «notAfter»

-- идентификатор и синтаксис расширения «политики сертификации»

id-ce-certificatePolicies       OBJECT IDENTIFIER ::=  { id-ce 32 }

anyPolicy                       OBJECT IDENTIFIER ::= { id-ce-certificatePolicies 0 }

CertificatePolicies ::=         SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::=           SEQUENCE {
    policyIdentifier                    CertPolicyId,
    policyQualifiers        SEQUENCE SIZE (1..MAX) OF
                                        PolicyQualifierInfo OPTIONAL }

CertPolicyId ::=                OBJECT IDENTIFIER
PolicyQualifierInfo ::=         SEQUENCE {
    policyQualifierId                   PolicyQualifierId,
    qualifier                   ANY DEFINED BY policyQualifierId }

-- прикладные системы и ИТС, которые распознают дополнительные определители
-- политик, обязаны вставить следующее определение «PolicyQualifierId»

PolicyQualifierId ::=           OBJECT IDENTIFIER ( id-qt-cps | id-qt-unotice )

-- определитель указателя на CPS (Certification Practice Statement)

CPSuri ::=                              IA5String

-- определитель уведомления пользователя («UserNotice»)

UserNotice ::=                  SEQUENCE {
    noticeRef                           NoticeReference     OPTIONAL,
    explicitText                        DisplayText         OPTIONAL }

NoticeReference ::=             SEQUENCE {
    Organization                        DisplayText,
    noticeNumbers               SEQUENCE OF INTEGER }

DisplayText ::=                 CHOICE {
    ia5String                           IA5String       (SIZE (1..200)),
    visibleString                       VisibleString   (SIZE (1..200)),
    bmpString                           BMPString       (SIZE (1..200)),
    utf8String                          UTF8String      (SIZE (1..200)) }

-- идентификатор и синтаксис расширения «отображения политики»

id-ce-policyMappings            OBJECT IDENTIFIER ::=  { id-ce 33 }

PolicyMappings ::=      SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    issuerDomainPolicy                  CertPolicyId,
    subjectDomainPolicy                 CertPolicyId }
```

```
-- идентификатор и синтаксис расширения «альтернативное имя владельца»

id-ce-subjectAltName            OBJECT IDENTIFIER ::= { id-ce 17 }

SubjectAltName ::=              GeneralNames

GeneralNames ::=                SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::=                 CHOICE {
    otherName                   [0]     AnotherName,
    rfc822Name                  [1]     IA5String,
    dNSName                     [2]     IA5String,
    x400Address                 [3]     ORAddress,
    directoryName               [4]     Name,
    ediPartyName                [5]     EDIPartyName,
    uniformResourceIdentifier   [6]     IA5String,
    iPAddress                   [7]     OCTET STRING,
    registeredID                [8]     OBJECT IDENTIFIER }

-- «AnotherName» заменяет «OTHER-NAME ::= TYPE-IDENTIFIER», так как
-- «TYPE-IDENTIFIER» не указан в синтаксисе ASN.1-стандарта версия 1988 г.

AnotherName ::=                 SEQUENCE {
    type-id                             OBJECT IDENTIFIER,
    value                       [0]     EXPLICIT ANY DEFINED BY type-id }

EDIPartyName ::=                SEQUENCE {
    nameAssigner                [0]     DirectoryString         OPTIONAL,
    partyName                   [1]     DirectoryString }

-- идентификатор и синтаксис расширения «альтернативное имя издателя»

id-ce-issuerAltName             OBJECT IDENTIFIER ::= { id-ce 18 }

IssuerAltName ::=               GeneralNames

id-ce-subjectDirectoryAttributes    OBJECT IDENTIFIER ::= { id-ce 9 }

SubjectDirectoryAttributes ::=  SEQUENCE SIZE (1..MAX) OF Attribute

-- идентификатор и синтаксис расширения «основные ограничения»

id-ce-basicConstraints          OBJECT IDENTIFIER ::= { id-ce 19 }

BasicConstraints ::=            SEQUENCE {
    cA                                  BOOLEAN DEFAULT FALSE,
    pathLenConstraint                   INTEGER (0..MAX)    OPTIONAL }

-- идентификатор и синтаксис расширения «ограничения на наименования»

id-ce-nameConstraints           OBJECT IDENTIFIER ::= { id-ce 30 }

NameConstraints ::=             SEQUENCE {
    permittedSubtrees           [0]     GeneralSubtrees     OPTIONAL,
    excludedSubtrees            [1]     GeneralSubtrees     OPTIONAL }

GeneralSubtrees ::=             SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::=          SEQUENCE {
    base                                GeneralName,
    minimum                 [0]         BaseDistance    DEFAULT 0,
    maximum                 [1]         BaseDistance    OPTIONAL }
```

```
BaseDistance ::=                INTEGER (0..MAX)

-- идентификатор и синтаксис расширения «ограничения политик»

id-ce-policyConstraints         OBJECT IDENTIFIER ::= { id-ce 36 }

PolicyConstraints ::=           SEQUENCE {
    requireExplicitPolicy       [0]     SkipCerts       OPTIONAL,
    inhibitPolicyMapping        [1]     SkipCerts       OPTIONAL }

SkipCerts ::=                   INTEGER (0..MAX)

-- идентификатор и синтаксис расширения «точки распространения СОС»

id-ce-cRLDistributionPoints     OBJECT IDENTIFIER  ::= {id-ce 31}

CRLDistributionPoints ::=       SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::=           SEQUENCE {
    distributionPoint           [0]     DistributionPointName  OPTIONAL,
    reasons                     [1]     ReasonFlags            OPTIONAL,
    cRLIssuer                   [2]     GeneralNames           OPTIONAL }

DistributionPointName ::=       CHOICE {
    fullName                    [0]     GeneralNames,
    nameRelativeToCRLIssuer     [1]     RelativeDistinguishedName }

ReasonFlags ::=                 BIT STRING {
    unused                      (0),
    keyCompromise               (1),
    cACompromise        (2),
    affiliationChanged          (3),
    superseded                  (4),
    cessationOfOperation        (5),
    certificateHold             (6),
    privilegeWithdrawn          (7),
    aACompromise        (8) }

-- идентификатор и синтаксис расширения «расширение области применения ключа»

id-ce-extKeyUsage       OBJECT IDENTIFIER ::= {id-ce 37}

ExtKeyUsageSyntax ::=           SEQUENCE SIZE (1..MAX) OF KeyPurposeId

KeyPurposeId ::=                OBJECT IDENTIFIER

-- разрешение на использование неустановленного ключа

anyExtendedKeyUsage             OBJECT IDENTIFIER ::= { id-ce-extKeyUsage 0 }

-- идентификаторы расширения области применения ключа

        id-kp-serverAuth            OBJECT IDENTIFIER ::= { id-kp 1 }
        id-kp-clientAuth        OBJECT IDENTIFIER ::= { id-kp 2 }
        id-kp-codeSigning           OBJECT IDENTIFIER ::= { id-kp 3 }
        id-kp-emailProtection       OBJECT IDENTIFIER ::= { id-kp 4 }
        id-kp-timeStamping          OBJECT IDENTIFIER ::= { id-kp 8 }
        id-kp-OCSPSigning           OBJECT IDENTIFIER ::= { id-kp 9 }

-- идентификатор и синтаксис «запрет на использование субполя «anyPolicy»

id-ce-inhibitAnyPolicy          OBJECT IDENTIFIER ::= { id-ce 54 }
```

```
InhibitAnyPolicy ::=                SkipCerts

-- идентификатор и синтаксис расширения «самый последний (усечённый) COC»

id-ce-freshestCRL                   OBJECT IDENTIFIER ::=  { id-ce 46 }

FreshestCRL ::=                     CRLDistributionPoints

-- доступ к информации УЦ

id-pe-authorityInfoAccess           OBJECT IDENTIFIER ::= { id-pe 1 }

AuthorityInfoAccessSyntax  ::=      SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=              SEQUENCE {
        accessMethod                        OBJECT IDENTIFIER,
        accessLocation              GeneralName  }

-- доступ к информации владельца сертификата

id-pe-subjectInfoAccess             OBJECT IDENTIFIER ::= { id-pe 11 }

SubjectInfoAccessSyntax  ::=        SEQUENCE SIZE (1..MAX) OF AccessDescription

-- идентификатор и синтаксис расширения «номер COC»

id-ce-cRLNumber                     OBJECT IDENTIFIER ::= { id-ce 20 }

CRLNumber ::=                       INTEGER (0..MAX)

-- идентификатор и синтаксис расширения «точка распространения COC»

id-ce-issuingDistributionPoint      OBJECT IDENTIFIER ::= { id-ce 28 }

IssuingDistributionPoint ::=        SEQUENCE {
        distributionPoint                   [0]     DistributionPointName   OPTIONAL,
        onlyContainsUserCerts       [1]     BOOLEAN DEFAULT FALSE,
        onlyContainsCACerts                 [2]     BOOLEAN DEFAULT FALSE,
        onlySomeReasons                     [3]     ReasonFlags             OPTIONAL,
        indirectCRL                         [4]     BOOLEAN DEFAULT FALSE,
        onlyContainsAttributeCerts          [5]     BOOLEAN DEFAULT FALSE }
        -- не более, чем одна из субпоследовательностей «onlyContainsUserCerts»,
        -- «onlyContainsCACerts», «onlyContainsAttributeCerts» может иметь
        -- значение «TRUE».

id-ce-deltaCRLIndicator             OBJECT IDENTIFIER ::= { id-ce 27 }

BaseCRLNumber ::=                   CRLNumber

-- идентификатор и синтаксис расширения «коды причины»

id-ce-cRLReasons            OBJECT IDENTIFIER ::= { id-ce 21 }

CRLReason ::=               ENUMERATED {
        Unspecified             (0),
        keyCompromise           (1),
        cACompromise (2),
        affiliationChanged      (3),
        superseded              (4),
        cessationOfOperation    (5),
        certificateHold         (6),
```

```
                removeFromCRL       (8),
                privilegeWithdrawn  (9),
                aACompromise (10) }

-- идентификатор и синтаксис расширения записи СОС «издатель сертификата»

id-ce-certificateIssuer             OBJECT IDENTIFIER ::= { id-ce 29 }

CertificateIssuer ::=               GeneralNames

-- идентификатор и синтаксис расширения «правила хранения»

id-ce-holdInstructionCode           OBJECT IDENTIFIER ::= { id-ce 23 }

HoldInstructionCode ::=             OBJECT IDENTIFIER

-- субветвь «holdinstruction» в ветви «ANSI x9» дерева иерархии
-- Интернет/PKI-инфраструктуры

holdInstruction                     OBJECT IDENTIFIER ::=
                                    {joint-iso-itu-t(2) member-body(2) us(840) x9cm(10040) 2}

-- «holdinstructions» стандарта ANSI X9

id-holdinstruction-none             OBJECT IDENTIFIER  ::=
                                    {holdInstruction 1} -- deprecated

id-holdinstruction-callissuer       OBJECT IDENTIFIER ::= {holdInstruction 2}

id-holdinstruction-reject           OBJECT IDENTIFIER ::= {holdInstruction 3}

-- идентификатор и синтаксис расширения записи СОС «недействительная дата»

id-ce-invalidityDate          OBJECT IDENTIFIER ::= { id-ce 24 }

InvalidityDate ::=                  GeneralizedTime

END
```

Приложение В. Комментарии к ASN.1-модулю

УЦ обязаны сделать всё, чтобы последовательный номер «*serialNumber*» был неотрицательным целым числом, т.е. бит знака («+» или «–») целого числа «*INTEGER*» в DER-кодировке всегда должен быть равен нулю. Это может быть сделано путём добавления старшего октета (с левой стороны) «00000000» («00» в шестнадцатеричном формате), если это необходимо. Это исключит потенциальную неопределённость при отображении последовательности октетов в целое число и наоборот.

Как указывалось в §4.1.2.2, последовательные номера могут быть очень длинными целыми числами. Пользователи сертификатов должны быть способны обрабатывать значение в субполе «*serialNumber*» длиной до 20 октетов

(включительно). Следующие этому стандарту УЦ не должны использовать значения в субполе «*serialNumber*» длиной более 20 октетов.

Как указывалось в §5.2.3,, номера СОС могут очень длинными целыми числами. Субъекты, проверяющие номера СОС, обязаны обрабатывать значения последовательности «*CRLNumber*» длиной до 20 октетов. Издатели СОС, придерживающиеся данного стандарта, обязаны не использовать значения последовательности «*CRLNumber*» длиной более 20 октетов.

Конструкция «*SEQUENCE SIZE (1..MAX) OF*» представлена в нескольких *ASN.1*-конструкциях. Допустимая *ASN.1*-последовательность может иметь ноль или несколько записей (строк). Конструкция «*SIZE (1..MAX)*» ограничивает последовательность, которая должна иметь, по крайней мере, одну запись. «*MAX*» указывает на то, что верхняя граница не установлена. Прикладные системы и ИТС не имеют ограничений на выбор верхней границы, которая удовлетворяет их требования.

Формат последовательности символов «*PrintableString*» включает только базовый набор латинских символов: буквы от «a» до «z» в нижнем регистре, буквы от «A» до «Z» в верхнем регистре, цифры от «0» до «9», одиннадцать специальных символов «‘», «=», «(», «)», «+» «,», «–» , «.», «/», «:», «?» и пробел.

Целесообразно, чтобы прикладные системы и ИТС обращали внимание на то, что символ «@» и знак подчёркивания «_» не поддерживаются в *PrintableString*-кодировке *ASN.1*-формата. Эти символы очень часто используются в адресах Интернет-сети. Такие адреса должны, в обязательном порядке, кодироваться с использованием *ASN.1*-формата, который поддерживает их. Обычно они кодируются с помощью *IA5String*-формата в качестве, либо атрибута почтового адреса «*emailAddress*» в рамках уникального имени, либо в *rfc822Name*-формате обобщённого имени «*GeneralName*». Прикладные системы и ИТС, придерживающиеся данного стандарта, обязаны не кодировать последовательности, содержащие символ «@» или знак подчёркивания «_», в *PrintableString*-формате. Последовательность символов в формате «*TeletexString*» является су-

пермножеством по отношению к *PrintableString*-формату. *TeletexString*-кодировка поддерживает, в некоторой степени, стандарт Латинского алфавита (подобно ASCII-коду): Латинские символы с ударениями без пробелов и Японские символы.

Перечни поименованных битов представляют собой формат битовых последовательностей «*BIT STRING*», в которых значениям были присвоены имена. Данный стандарт использует перечни поименованных битов в определениях субполей «*keyUsage*», «*cRLDistributionPoints*» и «*freshestCRL*» поля «Расширения» сертификата, а также последовательностей «*freshestCRL*» и «*issuingDistributionPoint*» субполя «*crlExtensions*» COC. Если используется DER-кодировка перечня поименованных бит, то завершающие нули должны, в обязательном порядке, отсутствовать. Т.е., закодированное значение заканчивается последним поименованным битом, имеющим значение «*1*».

Формат последовательности символов «*UniversalString*» включает любой из символов, разрешённых стандартом ISO 10646. Этот стандарт вводит Совокупность универсальных много-октетных кодовых символов (UCS-алфавит).

Формат последовательности символов «*UTF8String*» был введён *ASN.1*-стандартом (версия 1997 г.), а затем *UTF8String*-кодировка была включена в Рекомендацию ITU-T X.520 версии 2001 г. (в перечень выбираемых кодов для *DirectoryString*-формата). *UTF8String*-формат является универсальным и ему был присвоен кодовый номер «12». *UTF8String*-формат представлен в стандарте RFC-3629.

В большинстве форматов атрибутов, установленных Рекомендацией ITU-T X.520, значение атрибута «*AttributeValue*» использует *DirectoryString*-формата. Атрибуты, представленные в Приложении A, а именно «*name*», «*surname*», «*givenName*», «*initials*», «*generationQualifier*», «*commonName*», «*localityName*», «*stateOrProvinceName*», «*organizationName*», «*organizationalUnitName*», «*title*» и «*pseudonym*», используют *DirectoryString*-формат. Рекомендация ITU-T X.520 использует описание параметрического формата (Рекомендация ITU-T X.683) *DirectoryString*-кодировки с целью определения синтаксиса каждого из этих ат-

рибутов. А параметр используется для указания максимальной длины последовательности символов, допустимой для конкретного атрибута. В Приложении А, с целью исключения описаний параметрического формата, *DirectoryString*-формат записывается в своей расширенной форме для описания каждого из этих типов атрибутов. Таким же образом, *ASN.1*-стандарт в Приложении А определяет синтаксис каждого из этих атрибутов в качестве выбора («*CHOICE*») из *TeletexString*-, *PrintableString*-, *UniversalString*-, *UTF8String*- и *BMPString*-кодировок, с соответствующими ограничениями на длину последовательности, применяемыми для каждого из форматов в строке «*CHOICE*», а не использует *ASN.1*-формат *DirectoryString*-кодировки для описания синтаксиса.

Целесообразно, чтобы прикладные системы и ИТС обращали внимание на то, что DER-кодировка значений «*SET OF*» требует упорядочивания закодированных значений. Соответственно, та же проблема касается и уникальных имён.

Целесообразно, чтобы прикладные системы и ИТС обращали внимание на то, что DER-кодирование компонента «*SET*» или «*SEQUENCE*», значение которого — «*DEFAULT*», не включает компонент из закодированного СЕРТ или СОС. Например, субполе «*basicConstraints*» поля «Расширения» сертификата, в котором флаг «*cA*» имеет значение «*FALSE*», могло бы не включать флаг «*cA*» (тип данных, «*boolean*») из закодированного СЕРТ.

Идентификаторы объектов (OID) используются в данном стандарте для идентификации политик сертификации, алгоритмов открытых ключей и криптоалгоритмов, расширений СЕРТ и др. Для OID не существует максимальных значений. Данный стандарт даёт полномочия на обработку OID, которые включают маркеры (ветви в дереве иерархии OID), имеющие значения меньше чем 2^{28}, т.е. они должны быть, в обязательном порядке в диапазоне от 0 до 268435455, включительно. Это позволяет представлять каждый маркер в форме одного 32-битового слова. Кроме того, прикладные системы и ИТС обязаны обрабатывать OID, в которых длина последовательностей десятичных чисел, разделённых точками (§1.4 стандарта RFC-4512), может достигать до 100 байт,

включительно. Прикладные системы и ИТС обязаны обрабатывать OID, в которых длина достигает до 20 маркеров, включительно. Целесообразно, чтобы УЦ не издавали СЕРТ, которые содержат OID, превышающие эти требования. Таким же образом, целесообразно, чтобы издатели СОС не выпускали СОС, которые содержат OID, превышающие эти требования.

Содержание конкретных правил кодирования значений в формате «*GeneralName*» в субполе «*nameConstraints*» поля «*Расширения*» отличаются от правил, которые применяются в других расширениях. Во всех других расширениях сертификата, СОС и записи СОС, представленных в данном стандарте, правила кодирования соответствуют правилам базового формата. Например, значения последовательности «*uniformResourceIdentifier*» должны, в обязательном порядке, включать приемлемый URI-идентификатор, как это определено в стандарте RFC-3986. Содержание конкретных правил кодирования значений в субполе «*basicConstraints*» поля «Расширения» СЕРТ представлены в §4.2.1.10, а эти правила могут не соответствовать правилам для базового формата. Например, когда в субполе «*basicConstraints*» поля «Расширения» СЕРТ содержится последовательность «*uniformResourceIdentifier*», то она должна, обязательно, включать DNS-имя (например, «*host.example.com*» или «*.example.com*»), а не URI-идентификатор.

Разработчики прикладных систем и ИТС должны знать, что Рекомендации ITU-T серии X.500 устанавливают группы расширяемых правил. Эти правила определяют, когда *ASN.1*-описание (модуль) может быть изменено без присвоения нового OID. Например, по крайней мере, два описания расширений включены в стандарт RFC-2459, предшественник данного стандарта ввёл другие *ASN.1*-описания (модули) по сравнению с данным стандартом, но использует одни и те же OID. Если в расширение включены неизвестные элементы, а само расширение не помечено как критичное, то такие неизвестные элементы должны быть проигнорированы следующим образом:

а) игнорировать все неизвестные назначенные битовые имена в рамках битовой последовательности;

b) игнорировать все неизвестные поименованные числа в формате «*ENUMERATED*» или «*INTEGER*», который используется в одном из перечисленных названий, при условии, что число встречается как дополнительный элемент компонента «*SET*» или «*SEQUENCE*»;

c) игнорировать все неизвестные элементы в компонентах «*SET*», в конце компонентов «*SEQUENCE*» или в строке «*CHOICE*», если строка «*CHOICE*» сама является элементом компонента «*SET*» или «*SEQUENCE*».

Если расширение содержит непредусмотренные значения, помеченные как критичные, прикладные системы и ИТС обязаны удалять СЕРТ или СОС, включающий неустановленное расширение.

Приложение С. Примеры

В этом Приложении представлены четыре примера: три СЕРТ и СОС. Первые два сертификата и СОС включают минимальный МС.

В §С.1 представлена распечатка само-подписанного сертификата в шестнадцатеричном коде с комментариями, изданного УЦ, уникальное имя которого «*cn=Example CA,dc=example,dc=com*». Сертификат содержит открытый RSA-ключ и был подписан соответствующим закрытым RSA-ключом.

В §С.2 представлена распечатка сертификата конечного пользователя в шестнадцатеричном коде с комментариями. Сертификат конечного пользователя содержит открытый RSA-ключ и был подписан закрытым ключом, соответствующим само-подписанному сертификату (§С.1).

В §С.3 представлена распечатка сертификата конечного пользователя в шестнадцатеричном коде с комментариями, содержащего открытый DSA-ключ с параметрами. СЕРТ был подписан с помощью DSA- и SHA-1-алгоритмов. Этот сертификат не является частью минимального МС.

В §С.3 представлена распечатка СОС в шестнадцатеричном коде с комментариями. СОС был издан УЦ, уникальное имя которого «*cn=Example CA,dc=example,dc=com*», а перечень аннулированных сертификатов включает сертификат конечного пользователя, представленный в §С.2.

С целью получения выходных данных все СЕРТ были обработаны с использованием вспомогательной программы «*dumpasn1*» Питера Гутмана (Peter Gutmann). Программу «*dumpasn1*» можно найти по адресу: <http://www.cs.auckland.ac.nz/~pgut001/dumpasn1.c>. Двоичный код сертификатов и СОС можно найти по адресу: <http://csrc.nist.gov/groups/ST/crypto_apps_infra/documents/pkixtools>. В тех местах данного Приложения, где уникальное имя представлено в формате последовательности символов, последние форматировались с помощью правил, представленных в стандарте RFC-4514.

§C.1 Само-подписанный сертификат с использованием RSA-алгоритма

Далее представлена 578-байтовая распечатка сертификата 3-ей версии в шестнадцатеричном коде с комментариями. СЕРТ включает следующую информацию:

a) последовательный номер — 17;

b) СЕРТ подписан с помощью RSA- и SHA-1(хэш)-алгоритмов;

c) уникальное имя издателя: «*cn=Example CA,dc=example,dc=com*»;

d) уникальное имя владельца: «*cn=Example CA,dc=example,dc=com*»;

e) СЕРТ был выпущен 30 апреля 2004 года, а срок его действия истёк 30 апреля 2005 года;

f) СЕРТ содержит 1024-битовый открытый RSA-ключ;

g) СЕРТ содержит субполе «*subjectKeyIdentifier*» поля «Расширения», сформированное с использованием способа *1)*, представленного в §4.2.1.2;

h) СЕРТ является сертификатом УЦ (как указано в субполе «*basicConstraints*» поля «*Расширения*»).

```
0    574  : SEQUENCE {
4    423  :   SEQUENCE {
8      3  :     [0] {
10     1  :       INTEGER 2
          :       }
13     1  :     INTEGER 17
16    13  :     SEQUENCE {
18     9  :       OBJECT IDENTIFIER
          :         sha1withRSAEncryption (1 2 840 113549 1 1 5)
29     0  :       NULL
                  :     }
```

```
31    67 :   SEQUENCE {
33    19 :     SET {
35    17 :       SEQUENCE {
37    10 :         OBJECT IDENTIFIER
             :           domainComponent (0 9 2342 19200300 100 1 25)
49     3 :         IA5String 'com'
             :         }
             :       }
54    23 :     SET {
56    21 :       SEQUENCE {
58    10 :         OBJECT IDENTIFIER
             :           domainComponent (0 9 2342 19200300 100 1 25)
70     7 :         IA5String 'example'
             :         }
             :       }
79    19 :     SET {
81    17 :       SEQUENCE {
83     3 :         OBJECT IDENTIFIER commonName (2 5 4 3)
88    10 :         PrintableString 'Example CA'
             :         }
             :       }
             :     }
100   30 :   SEQUENCE {
102   13 :     UTCTime 30/04/2004 14:25:34 GMT
117   13 :     UTCTime 30/04/2005 14:25:34 GMT
             :     }
132   67 :   SEQUENCE {
134   19 :     SET {
136   17 :       SEQUENCE {
138   10 :         OBJECT IDENTIFIER
             :           domainComponent (0 9 2342 19200300 100 1 25)
150    3 :         IA5String 'com'
             :         }
             :       }
155   23 :     SET {
157   21 :       SEQUENCE {
159   10 :         OBJECT IDENTIFIER
             :           domainComponent (0 9 2342 19200300 100 1 25)
171    7 :         IA5String 'example'
             :         }
             :       }
180   19 :     SET {
182   17 :       SEQUENCE {
184    3 :         OBJECT IDENTIFIER commonName (2 5 4 3)
189   10 :         PrintableString 'Example CA'
             :         }
             :       }
             :     }
201  159 :   SEQUENCE {
204   13 :     SEQUENCE {
206    9 :       OBJECT IDENTIFIER
             :         rsaEncryption (1 2 840 113549 1 1 1)
217    0 :       NULL
             :       }
219  141 :     BIT STRING, encapsulates {
223  137 :       SEQUENCE {
226  129 :         INTEGER
             :           00 C2 D7 97 6D 28 70 AA 5B CF 23 2E 80 70 39 EE
             :           DB 6F D5 2D D5 6A 4F 7A 34 2D F9 22 72 47 70 1D
             :           EF 80 E9 CA 30 8C 00 C4 9A 6E 5B 45 B4 6E A5 E6
             :           6C 94 0D FA 91 E9 40 FC 25 9D C7 B7 68 19 56 8F
             :           11 70 6A D7 F1 C9 11 4F 3A 7E 3F 99 8D 6E 76 A5
```

```
                :          74 5F 5E A4 55 53 E5 C7 68 36 53 C7 1D 3B 12 A6
                :          85 FE BD 6E A1 CA DF 35 50 AC 08 D7 B9 B4 7E 5C
                :          FE E2 A3 2C D1 23 84 AA 98 C0 9B 66 18 9A 68 47
                :          E9
358    3        :        INTEGER 65537
                :        }
                :      }
                :    }
363   66        :    [3] {
365   64        :      SEQUENCE {
367   29        :        SEQUENCE {
369    3        :          OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
374   22        :          OCTET STRING, encapsulates {
376   20        :            OCTET STRING
                :              08 68 AF 85 33 C8 39 4A 7A F8 82 93 8E 70 6A 4A
                :              20 84 2C 32
                :            }
                :          }
398   14        :        SEQUENCE {
400    3        :          OBJECT IDENTIFIER keyUsage (2 5 29 15)
405    1        :          BOOLEAN TRUE
408    4        :          OCTET STRING, encapsulates {
410    2        :            BIT STRING 1 unused bits
                :              '0000011'B
                :            }
                :          }
414   15        :        SEQUENCE {
416    3        :          OBJECT IDENTIFIER basicConstraints (2 5 29 19)
421    1        :          BOOLEAN TRUE
424    5        :          OCTET STRING, encapsulates {
426    3        :            SEQUENCE {
428    1        :              BOOLEAN TRUE
                :              }
                :            }
                :          }
                :        }
                :      }
                :    }
431   13        :  SEQUENCE {
433    9        :    OBJECT IDENTIFIER
                :      sha1withRSAEncryption (1 2 840 113549 1 1 5)
444    0        :    NULL
                :    }
446  129        :  BIT STRING
                :    6C F8 02 74 A6 61 E2 64 04 A6 54 0C 6C 72 13 AD
                :    3C 47 FB F6 65 13 A9 85 90 33 EA 76 A3 26 D9 FC
                :    D1 0E 15 5F 28 B7 EF 93 BF 3C F3 E2 3E 7C B9 52
                :    FC 16 6E 29 AA E1 F4 7A 6F D5 7F EF B3 95 CA F3
                :    66 88 83 4E A1 35 45 84 CB BC 9B B8 C8 AD C5 5E
                :    46 D9 0B 0E 8D 80 E1 33 2B DC BE 2B 92 7E 4A 43
                :    A9 6A EF 8A 63 61 B3 6E 47 38 BE E8 0D A3 67 5D
                :    F3 FA 91 81 3C 92 BB C5 5F 25 25 EB 7C E7 D8 A1
                :  }
```

§C.2 Сертификат конечного пользователя, изданный с использованием RSA-алгоритма

Далее представлена 629-байтовая распечатка сертификата 3-ей версии в шестнадцатеричном коде с комментариями. CEPT включает следующую информацию:

a) последовательный номер — 18;

b) CEPT подписан с помощью RSA- и SHA-1(хэш)-алгоритмов;

c) уникальное имя издателя: «*cn=Example CA,dc=example,dc=com*»;

d) уникальное имя владельца: «*cn= End Entity,dc=example,dc=com*»;

e) срок действия CEPT с 15 сентября 2004 года по 15 марта 2005 года;

f) CEPT содержит 1024-битовый открытый RSA-ключ;

g) CEPT является сертификатом конечного пользователя (так как субполе «*basicConstraints*» в поле «*Расширения*» не представлено);

h) CEPT содержит идентификатор «*authorityKeyIdentifier*», который совпадает с идентификатором «*subjectKeyIdentifier*» из CEPT, представленного в §C.1;

i) CEPT включает альтернативное имя — адрес электронной почтовой службы («*rfc822Name*») — «*end.entity@example.com*».

```
0     625  : SEQUENCE {
4     474  :   SEQUENCE {
8       3  :     [0] {
10      1  :       INTEGER 2
        :       }
13      1  :     INTEGER 18
16     13  :     SEQUENCE {
18      9  :       OBJECT IDENTIFIER
        :         sha1withRSAEncryption (1 2 840 113549 1 1 5)
29      0  :       NULL
        :       }
31     67  :     SEQUENCE {
33     19  :       SET {
35     17  :         SEQUENCE {
37     10  :           OBJECT IDENTIFIER
        :             domainComponent (0 9 2342 19200300 100 1 25)
49      3  :           IA5String 'com'
        :           }
        :         }
54     23  :       SET {
56     21  :         SEQUENCE {
58     10  :           OBJECT IDENTIFIER
        :             domainComponent (0 9 2342 19200300 100 1 25)
70      7  :           IA5String 'example'
        :           }
        :         }
79     19  :       SET {
81     17  :         SEQUENCE {
```

```
 83     3 :       OBJECT IDENTIFIER commonName (2 5 4 3)
 88    10 :       PrintableString 'Example CA'
          :       }
          :     }
          :   }
100    30 :   SEQUENCE {
102    13 :     UTCTime 15/09/2004 11:48:21 GMT
117    13 :     UTCTime 15/03/2005 11:48:21 GMT
          :     }
132    67 :   SEQUENCE {
134    19 :     SET {
136    17 :       SEQUENCE {
138    10 :         OBJECT IDENTIFIER
          :           domainComponent (0 9 2342 19200300 100 1 25)
150     3 :         IA5String 'com'
          :         }
          :       }
155    23 :     SET {
157    21 :       SEQUENCE {
159    10 :         OBJECT IDENTIFIER
          :           domainComponent (0 9 2342 19200300 100 1 25)
171     7 :         IA5String 'example'
          :         }
          :       }
180    19 :     SET {
182    17 :       SEQUENCE {
184     3 :         OBJECT IDENTIFIER commonName (2 5 4 3)
189    10 :         PrintableString 'End Entity'
          :         }
          :       }
          :     }
201   159 :   SEQUENCE {
204    13 :     SEQUENCE {
206     9 :       OBJECT IDENTIFIER
          :         rsaEncryption (1 2 840 113549 1 1 1)
217     0 :       NULL
          :       }
219   141 :     BIT STRING, encapsulates {
223   137 :       SEQUENCE {
226   129 :         INTEGER
          :           00 E1 6A E4 03 30 97 02 3C F4 10 F3 B5 1E 4D 7F
          :           14 7B F6 F5 D0 78 E9 A4 8A F0 A3 75 EC ED B6 56
          :           96 7F 88 99 85 9A F2 3E 68 77 87 EB 9E D1 9F C0
          :           B4 17 DC AB 89 23 A4 1D 7E 16 23 4C 4F A8 4D F5
          :           31 B8 7C AA E3 1A 49 09 F4 4B 26 DB 27 67 30 82
          :           12 01 4A E9 1A B6 C1 0C 53 8B 6C FC 2F 7A 43 EC
          :           33 36 7E 32 B2 7B D5 AA CF 01 14 C6 12 EC 13 F2
          :           2D 14 7A 8B 21 58 14 13 4C 46 A3 9A F2 16 95 FF
          :           23
358     3 :         INTEGER 65537
          :         }
          :       }
          :     }
363   117 :   [3] {
365   115 :     SEQUENCE {
367    33 :       SEQUENCE {
369     3 :         OBJECT IDENTIFIER subjectAltName (2 5 29 17)
374    26 :         OCTET STRING, encapsulates {
376    24 :           SEQUENCE {
378    22 :             [1] 'end.entity@example.com'
          :             }
          :           }
```

```
          :         }
402    29 :       SEQUENCE {
404     3 :         OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
409    22 :         OCTET STRING, encapsulates {
411    20 :           OCTET STRING
          :             17 7B 92 30 FF 44 D6 66 E1 90 10 22 6C 16 4F C0
          :             8E 41 DD 6D
          :           }
          :         }
433    31 :       SEQUENCE {
435     3 :         OBJECT IDENTIFIER
          :           authorityKeyIdentifier (2 5 29 35)
440    24 :         OCTET STRING, encapsulates {
442    22 :           SEQUENCE {
444    20 :             [0]
          :               08 68 AF 85 33 C8 39 4A 7A F8 82 93 8E 70 6A
          :               4A 20 84 2C 32
          :             }
          :           }
          :         }
466    14 :       SEQUENCE {
468     3 :         OBJECT IDENTIFIER keyUsage (2 5 29 15)
473     1 :         BOOLEAN TRUE
476     4 :         OCTET STRING, encapsulates {
478     2 :           BIT STRING 6 unused bits
          :             '11'B
          :           }
          :         }
          :       }
          :     }
          :   }
482    13 :   SEQUENCE {
484     9 :     OBJECT IDENTIFIER
          :       sha1withRSAEncryption (1 2 840 113549 1 1 5)
495     0 :     NULL
          :     }
497   129 :   BIT STRING
          :     00 20 28 34 5B 68 32 01 BB 0A 36 0E AD 71 C5 95
          :     1A E1 04 CF AE AD C7 62 14 A4 1B 36 31 C0 E2 0C
          :     3D D9 1E C0 00 DC 10 A0 BA 85 6F 41 CB 62 7A B7
          :     4C 63 81 26 5E D2 80 45 5E 33 E7 70 45 3B 39 3B
          :     26 4A 9C 3B F2 26 36 69 08 79 BB FB 96 43 77 4B
          :     61 8B A1 AB 91 64 E0 F3 37 61 3C 1A A3 A4 C9 8A
          :     B2 BF 73 D4 4D E4 58 E4 62 EA BC 20 74 92 86 0E
          :     CE 84 60 76 E9 73 BB C7 85 D3 91 45 EA 62 5D CD
          :   }
```

§C.3 Сертификат конечного пользователя, изданный с использованием DSA-алгоритма

Далее представлена 914-байтовая распечатка сертификата 3-ей версии в шестнадцатеричном коде с комментариями. CEPT включает следующую информацию:

a) последовательный номер — 256;

b) CEPT подписан с помощью DSA- и SHA-1(хэш)-алгоритмов;

c) уникальное имя издателя: «*cn=Example DSA CA,dc=example,dc=com*»;

d) уникальное имя владельца: «*cn=DSA End Entity,dc=example,dc=com*»;

e) срок действия СЕРТ с 2 мая 2004 года по 2 мая 2005 года;

f) СЕРТ содержит 1024-битовый открытый DSA-ключ с параметрами;

g) СЕРТ является сертификатом конечного пользователя (не СЕРТ УЦ);

h) СЕРТ включает альтернативное имя владельца — «*http://www.example.-com/users/DSAendentity.html*», и альтернативное имя издателя — «*http://www.-example.com*», и оба являются URL-указателями;

i) СЕРТ содержит идентификатор «*authorityKeyIdentifier*» и субполе «*certificatePolicies*» в поле «*Расширения*», которое определяет OID политики «*2.16.840.1.101.3.2.1.48.9*»;

j) СЕРТ содержит критичное субполе «*keyUsage*» в поле «Расширения», которое указывает на то, что открытый ключ предназначен для проверки ЭЦП.

```
0     910 : SEQUENCE {
4     846 :   SEQUENCE {
8       3 :     [0] {
10      1 :       INTEGER 2
        : :     }
13      2 :     INTEGER 256
17      9 :     SEQUENCE {
19      7 :       OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
        : :     }
28     71 :     SEQUENCE {
30     19 :       SET {
32     17 :         SEQUENCE {
34     10 :           OBJECT IDENTIFIER
        :             domainComponent (0 9 2342 19200300 100 1 25)
46      3 :           IA5String 'com'
        :           }
        :         }
51     23 :       SET {
53     21 :         SEQUENCE {
55     10 :           OBJECT IDENTIFIER
        :             domainComponent (0 9 2342 19200300 100 1 25)
67      7 :           IA5String 'example'
        :           }
        :         }
76     23 :       SET {
78     21 :         SEQUENCE {
80      3 :           OBJECT IDENTIFIER commonName (2 5 4 3)
85     14 :           PrintableString 'Example DSA CA'
        :           }
        :         }
        :       }
101    30 :     SEQUENCE {
103    13 :       UTCTime 02/05/2004 16:47:38 GMT
118    13 :       UTCTime 02/05/2005 16:47:38 GMT
```

```
          :     }
133    71 :    SEQUENCE {
135    19 :     SET {
137    17 :      SEQUENCE {
139    10 :       OBJECT IDENTIFIER
          :         domainComponent (0 9 2342 19200300 100 1 25)
151     3 :       IA5String 'com'
          :       }
          :      }
156    23 :     SET {
158    21 :      SEQUENCE {
160    10 :       OBJECT IDENTIFIER
          :         domainComponent (0 9 2342 19200300 100 1 25)
172     7 :       IA5String 'example'
          :       }
          :      }
181    23 :     SET {
183    21 :      SEQUENCE {
185     3 :       OBJECT IDENTIFIER commonName (2 5 4 3)
190    14 :       PrintableString 'DSA End Entity'
          :       }
          :      }
          :     }
206   439 :    SEQUENCE {
210   300 :     SEQUENCE {
214     7 :      OBJECT IDENTIFIER dsa (1 2 840 10040 4 1)
223   287 :      SEQUENCE {
227   129 :       INTEGER
          :         00 B6 8B 0F 94 2B 9A CE A5 25 C6 F2 ED FC FB 95
          :         32 AC 01 12 33 B9 E0 1C AD 90 9B BC 48 54 9E F3
          :         94 77 3C 2C 71 35 55 E6 FE 4F 22 CB D5 D8 3E 89
          :         93 33 4D FC BD 4F 41 64 3E A2 98 70 EC 31 B4 50
          :         DE EB F1 98 28 0A C9 3E 44 B3 FD 22 97 96 83 D0
          :         18 A3 E3 BD 35 5B FF EE A3 21 72 6A 7B 96 DA B9
          :         3F 1E 5A 90 AF 24 D6 20 F0 0D 21 A7 D4 02 B9 1A
          :         FC AC 21 FB 9E 94 9E 4B 42 45 9E 6A B2 48 63 FE
          :         43
359    21 :       INTEGER
          :         00 B2 0D B0 B1 01 DF 0C 66 24 FC 13 92 BA 55 F7
          :         7D 57 74 81 E5
382   129 :       INTEGER
          :         00 9A BF 46 B1 F5 3F 44 3D C9 A5 65 FB 91 C0 8E
          :         47 F1 0A C3 01 47 C2 44 42 36 A9 92 81 DE 57 C5
          :         E0 68 86 58 00 7B 1F F9 9B 77 A1 C5 10 A5 80 91
          :         78 51 51 3C F6 FC FC CC 46 C6 81 78 92 84 3D F4
          :         93 3D 0C 38 7E 1A 5B 99 4E AB 14 64 F6 0C 21 22
          :         4E 28 08 9C 92 B9 66 9F 40 E8 95 F6 D5 31 2A EF
          :         39 A2 62 C7 B2 6D 9E 58 C4 3A A8 11 81 84 6D AF
          :         F8 B4 19 B4 C2 11 AE D0 22 3B AA 20 7F EE 1E 57
          :         18
          :       }
          :      }
514   132 :     BIT STRING, encapsulates {
518   128 :      INTEGER
          :       30 B6 75 F7 7C 20 31 AE 38 BB 7E 0D 2B AB A0 9C
          :       4B DF 20 D5 24 13 3C CD 98 E5 5F 6C B7 C1 BA 4A
          :       BA A9 95 80 53 F0 0D 72 DC 33 37 F4 01 0B F5 04
          :       1F 9D 2E 1F 62 D8 84 3A 9B 25 09 5A 2D C8 46 8E
          :       2B D4 F5 0D 3B C7 2D C6 6C B9 98 C1 25 3A 44 4E
          :       8E CA 95 61 35 7C CE 15 31 5C 23 13 1E A2 05 D1
          :       7A 24 1C CB D3 72 09 90 FF 9B 9D 28 C0 A1 0A EC
          :       46 9F 0D B8 D0 DC D0 18 A6 2B 5E F9 8F B5 95 BE
```

```
                :       }
                :     }
649   202       :   [3] {
652   199       :     SEQUENCE {
655    57       :       SEQUENCE {
657     3       :         OBJECT IDENTIFIER subjectAltName (2 5 29 17)
662    50       :         OCTET STRING, encapsulates {
664    48       :           SEQUENCE {
666    46       :             [6]
                :               'http://www.example.com/users/DSAendentity.'
                :               'html'
                :             }
                :           }
                :         }
714    33       :       SEQUENCE {
716     3       :         OBJECT IDENTIFIER issuerAltName (2 5 29 18)
721    26       :         OCTET STRING, encapsulates {
723    24       :           SEQUENCE {
725    22       :             [6] 'http://www.example.com'
                :             }
                :           }
                :         }
749    29       :       SEQUENCE {
751     3       :         OBJECT IDENTIFIER subjectKeyIdentifier (2 5 29 14)
756    22       :         OCTET STRING, encapsulates {
758    20       :           OCTET STRING
                :             DD 25 66 96 43 AB 78 11 43 44 FE 95 16 F9 D9 B6
                :             B7 02 66 8D
                :           }
                :         }
780    31       :       SEQUENCE {
782     3       :         OBJECT IDENTIFIER
                :           authorityKeyIdentifier (2 5 29 35)
787    24       :         OCTET STRING, encapsulates {
789    22       :           SEQUENCE {
791    20       :             [0]
                :               86 CA A5 22 81 62 EF AD 0A 89 BC AD 72 41 2C
                :               29 49 F4 86 56
                :             }
                :           }
                :         }
813    23       :       SEQUENCE {
815     3       :         OBJECT IDENTIFIER certificatePolicies (2 5 29 32)
820    16       :         OCTET STRING, encapsulates {
822    14       :           SEQUENCE {
824    12       :             SEQUENCE {
826    10       :               OBJECT IDENTIFIER '2 16 840 1 101 3 2 1 48 9'
                :               }
                :             }
                :           }
                :         }
838    14       :       SEQUENCE {
840     3       :         OBJECT IDENTIFIER keyUsage (2 5 29 15)
845     1       :         BOOLEAN TRUE
848     4       :         OCTET STRING, encapsulates {
850     2       :           BIT STRING 7 unused bits
                :             '1'B (bit 0)
                :           }
                :         }
                :       }
                :     }
                :   }
```

```
854     9  :  SEQUENCE {
856     7  :    OBJECT IDENTIFIER dsaWithSha1 (1 2 840 10040 4 3)
           :    }
865    47  :  BIT STRING, encapsulates {
868    44  :    SEQUENCE {
870    20  :      INTEGER
           :        65 57 07 34 DD DC CA CC 5E F4 02 F4 56 42 2C 5E
           :        E1 B3 3B 80
892    20  :      INTEGER
           :        60 F4 31 17 CA F4 CF FF EE F4 08 A7 D9 B2 61 BE
           :        B1 C3 DA BF
           :      }
           :    }
           :  }
```

§C.4 COC

Далее представлена распечатка COC 2-ой версии в шестнадцатеричном коде с комментариями, включающего два расширения («*cRLNumber*» и «*authorityKeyIdentifier*»). COC был издан УЦ «*cn=Example CA,dc=example,dc=com*» 5 февраля 2005 года, а следующий выпуск планируется 5 февраля 2005 года. COC содержит один аннулированный сертификат: последовательный номер 18, который был отозван 19 ноября 2004 года в следствие компрометации ключа (код причины «*keyCompromise*»). Сам COC имеет номер 12 и был подписан с помощью RSA- и SHA-1-алгоритмов

```
0     352  : SEQUENCE {
4     202  :   SEQUENCE {
7       1  :     INTEGER 1
10     13  :     SEQUENCE {
12      9  :       OBJECT IDENTIFIER
           :         sha1withRSAEncryption (1 2 840 113549 1 1 5)
23      0  :       NULL
           :       }
25     67  :     SEQUENCE {
27     19  :       SET {
29     17  :         SEQUENCE {
31     10  :           OBJECT IDENTIFIER
           :             domainComponent (0 9 2342 19200300 100 1 25)
43      3  :           IA5String 'com'
           :           }
           :         }
48     23  :       SET {
50     21  :         SEQUENCE {
52     10  :           OBJECT IDENTIFIER
           :             domainComponent (0 9 2342 19200300 100 1 25)
64      7  :           IA5String 'example'
           :           }
           :         }
73     19  :       SET {
75     17  :         SEQUENCE {
77      3  :           OBJECT IDENTIFIER commonName (2 5 4 3)
```

```
 82    10 :         PrintableString 'Example CA'
          :         }
          :       }
          :     }
          :   }
 94    13 :   UTCTime 05/02/2005 12:00:00 GMT
109    13 :   UTCTime 06/02/2005 12:00:00 GMT
124    34 :   SEQUENCE {
126    32 :     SEQUENCE {
128     1 :       INTEGER 18
131    13 :       UTCTime 19/11/2004 15:57:03 GMT
146    12 :       SEQUENCE {
148    10 :         SEQUENCE {
150     3 :           OBJECT IDENTIFIER cRLReason (2 5 29 21)
155     3 :           OCTET STRING, encapsulates {
157     1 :             ENUMERATED 1
          :             }
          :           }
          :         }
          :       }
          :     }
160    47 :   [0] {
162    45 :     SEQUENCE {
164    31 :       SEQUENCE {
166     3 :         OBJECT IDENTIFIER
          :           authorityKeyIdentifier (2 5 29 35)
171    24 :         OCTET STRING, encapsulates {
173    22 :           SEQUENCE {
175    20 :             [0]
          :               08 68 AF 85 33 C8 39 4A 7A F8 82 93 8E 70 6A
          :               4A 20 84 2C 32
          :             }
          :           }
          :         }
197    10 :       SEQUENCE {
199     3 :         OBJECT IDENTIFIER cRLNumber (2 5 29 20)
204     3 :         OCTET STRING, encapsulates {
206     1 :           INTEGER 12
          :           }
          :         }
          :       }
          :     }
          :   }
209    13 :   SEQUENCE {
211     9 :     OBJECT IDENTIFIER
          :       sha1withRSAEncryption (1 2 840 113549 1 1 5)
222     0 :     NULL
          :     }
224   129 :   BIT STRING
          :     22 DC 18 7D F7 08 CE CC 75 D0 D0 6A 9B AD 10 F4
          :     76 23 B4 81 6E B5 6D BE 0E FB 15 14 6C C8 17 6D
          :     1F EE 90 17 A2 6F 60 E4 BD AA 8C 55 DE 8E 84 6F
          :     92 F8 9F 10 12 27 AF 4A D4 2F 85 E2 36 44 7D AA
          :     A3 4C 25 38 15 FF 00 FD 3E 7E EE 3D 26 12 EB D8
          :     E7 2B 62 E2 2B C3 46 80 EF 78 82 D1 15 C6 D0 9C
          :     72 6A CB CE 7A ED 67 99 8B 6E 70 81 7D 43 42 74
          :     C1 A6 AF C1 55 17 A2 33 4C D6 06 98 2B A4 FC 2E
          :   }
```