Network Working Group
Request for Comments: 4210
Obsoletes: 2510
Category: Standards Track

C. Adams
University of Ottawa
S. Farrell
Trinity College Dublin
T. Kause
SSH
T. Mononen
SafeNet

September 2005

Протокол управления сертификатами в инфраструктуре открытых ключей Internet X.509

Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)

Статус документа

Этот документ содержит проект стандартного протокола Internet для сообщества Internet и служит приглашением к дискуссии в целях развития и совершенствования протокола. Информацию о текущем состоянии стандартизации протокола можно найти в документе «Internet Official Protocol Standards» (STD 1). Документ можно распространять без ограничений.

Авторские права

Аннотация

В этом документе описан протокол управления сертификатами (CMP[1]) в инфраструктуре открытых ключей (PKI[2]) Internet X.509. Определены протокольные сообщения для создания сертификатов X.509v3 и управления ими. Протокол CMP обеспечивает интерактивное взаимодействие между компонентами PKI, включая обмен данными между агентами сертификации (CA[3]) и клиентскими системами.

Оглавление

[1]Certificate Management Protocol.
[2]Public Key Infrastructure.
[3]Certification Authority.

1. Введение

В этом документе описан протокол управления сертификатами (CMP[1]) в инфраструктуре открытых ключей (PKI[2]) Internet X.509. Определены протокольные сообщения для создания сертификатов X.509v3 и управления ими. Термин «сертификат» в данном документе обозначает сертификат X.509v3 в соответствии с определением [X509].

Данная спецификация отменяет документ RFC 2510 и имеет ряд отличий от RFC 2510:

- Раздел описания профиля управления PKI разделен на два приложения - обязательный профиль и дополнительный профиль. Некоторые из обязательных прежде функций перенесены в опциональный профиль.
- Существенно изменён механизм подтверждения сообщений.
- Введён новый механизм опроса, запрещающий старый метод опроса на транспортном уровне CMP.
- Транспортный протокол CMP вынесен в отдельный документ [CMPtrans] и соответствующий раздел удалён из данной спецификации.
- Введён новый метод явного подтверждения для снижения числа сообщений в одной транзакции..
- В новой спецификации в протокол внесены незначительные усовершенствования, а также прояснены некоторые детали описания.

2. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с [RFC2119].

3. Обзор управления PKI

Инфраструктура PKI должна быть структурирована в соответствии с типами её администраторов. Поддержка администраторов с неограниченным выбором не только усложняет реализацию программ, но и повышает вероятность того, что незначительные ошибки администраторов или разработчиков программ смогут вызывать существенные проблемы. Кроме того, использование громоздких механизмов ограничения возможностей администраторов будет приводить к отказу от использования PKI.

Протоколы управления **требуется** для поддержки интерактивного взаимодействия между компонентами PKI. Например, протокол управления может использоваться при взаимодействии между агентом сертификации (CA) и клиентской системой, с которой связана пара ключей, или между двумя CA, связанных отношениями кросс-сертификации.

3.1. Модель управления PKI

Прежде, чем задать какие-либо форматы сообщений или процедуры, определим элементы, вовлечённые в управление PKI и взаимодействие между этими элементами (в терминах требуемых функций управления PKI). После этого сгруппируем эти функции для создания различных определяемых типов конечных элементов.

[1]Certificate Management Protocol.
[2]Public Key Infrastructure.

3.1.1. Определения элементов PKI

Элементы, вовлечённые в управление PKI, включают конечные элементы (т. е., элементы, для которых выпускаются сертификаты) и агенты сертификации (элементы, выдающие сертификаты). В управлении PKI **может** также участвовать агент регистрации.

3.1.1.1. Субъекты и конечные элементы

Термин «субъект» в этом документе обозначает элементы, для которых выдаются (вводятся) сертификаты. Обычно имя этого элемента указывается в поле сертификата subject или subjectAltName. Когда нужно различать инструменты и/или программы, используемые субъектом (например, локальный модуль управления сертификатами), будем называть их «оборудованием субъекта». В общем случае вместо термина «субъект» предпочтительно использовать термин «конечный элемент» (EE[1]), во избежание путаницы с именем поля. Важно отметить, что в данном случае к конечному элементу относится не только человек, использующий программные приложения, но и сами такие приложения (например, средства защиты IP). Это оказывает влияние на протокол, используемый для работы управления PKI, - например, прикладные программы обычно знают более точно, чем пользователь-человек, какие расширения сертификатов нужны. Элементы управления PKI являются также конечными элементами в том смысле, что они иногда указываются в полях subject или subjectAltName сертификатов или кросс-сертификатов. Там, где это возможно, термин «конечный элемент» будет относиться к элементам, не являющимся элементами управления PKI.

Все конечные элементы требуют локальной защиты доступа к той или иной информации - по крайней мере, имя и секретный ключ элемента, имя CA, которому данный элемент доверяет напрямую, а также открытый ключ этого CA (или оттиск открытого ключа в тех случаях, когда доступна самосертифицированная версия). Разработчики **могут** использовать защищённое локальное хранилище, обеспечивающее хранение дополнительных данных (например, сертификат владельца конечного элемента или специфические данные приложения). Форма защищённого хранилища может меняться от обычных файлов до защищённых от копирования маркеров (брелоков). Информацию, находящуюся в таких защищённых локальных хранилищах, будем называть персональной средой защиты (PSE[2]) конечного элемента.

Хотя форматы PSE не рассматриваются в этом документе (они существенно зависят от типа оборудования и других параметров), базовый формат обмена PSE определён в этой спецификации - **может** использоваться сертификационный отклик.

3.1.1.2. Агент сертификации

Агент сертификации (CA) с точки зрения конечного элемента может быть (или не быть) «третьей стороной». Достаточно часто на практике CA относится к той же организации, которой принадлежат поддерживаемые агентом конечные элементы.

Термин «CA» будет использоваться для обозначения элементов, указываемых в поле эмитента (issuer) сертификата. Говоря о программах или аппаратных компонентах, используемых CA, будем применять термин «оборудование CA».

Оборудование CA зачастую будет включать «подключаемые» (off-line) и «постоянно включённые» (on-line) компоненты. Секретный ключ CA может размещаться только на подключаемых компонентах. Этот вопрос относится к компетенции разработчиков (хотя он связан также в политикой безопасности).

Термин «корневой CA[3]» (корневой агент сертификации будет использоваться применительно к CA, которым конечный элемент доверяет напрямую - для безопасной доставки открытого ключа корневого CA требуется использование отдельного «канала» (пути доставки). Это не означает, что корневой агент CA обязательно должен размещаться на вершине иерархии CA - это лишь агент, которому верят без дополнительных подтверждений (непосредственно).

«Второстепенными[4]» (подчинёнными) агентами CA будем называть агенты, которые не являются корневыми CA для рассматриваемого конечного элемента. Зачастую второстепенные CA не являются корневыми агентами ни для какого элемента, но это не обязательно.

3.1.1.3. Агент регистрации

В дополнение к конечным элементам и агентам CA во многих средах могут использоваться агенты регистрации (RA5), отличающиеся от CA. Выполняемые такими агентами функции зачастую существенно различаются, но **могут** включать персональную идентификацию, распространение маркеров (token), отчёты об отзыве (сертификатов), выделение имен, генерацию ключей, архивирование ключевых пар и т. п..

В этом документе RA считаются **опциональными** компонентами. При отсутствии агента регистрации предполагается, что все функции RA выполняет CA, поэтому протоколы управления PKI совпадают с точки зрения конечного элемента.

Используемые агентами регистрации средства будем называть «оборудованием RA».

Отметим, что сам агент RA является конечным элементом. В дальнейшем предполагается, что все RA на практике являются сертифицированными конечными элементами и имеют приватные ключи, которые могут служить для подписи. Вопрос идентификации оборудованием CA некоторых конечных элементов в качестве агентов регистрации отдаётся на откуп разработчиков (т. е., в этом документе не рассматриваются вопросы отдельной сертификации RA). Мы не требуем, чтобы RA сертифицировались агентами CA, с которыми они взаимодействуют в данный момент (один агент RA может работать с множеством CA, имея единственный сертификат).

При некоторых условиях конечные элементы будут напрямую взаимодействовать с CA даже при наличии RA. Например, для начальной регистрации и/или сертификации субъект может использовать RA, но напрямую взаимодействовать с CA для обновления своего сертификата.

---

[1] End entity.
[2] Personal Security Environment.
[3] Root CA.
[4] Subordinate CA.

3.1.2. Требования к управлению PKI

Описанный здесь протокол соответствует перечисленным ниже требованиям по управлению PKI.

1. Управление PKI должно соответствовать стандартам ISO/IEC 9594-8/ITU-T X.509.
2. Должна обеспечиваться возможность регулярного обновления любой пары ключей без воздействия на другие ключевые пары.
3. Защита конфиденциальности в протоколе управления PKI должна быть сведена к минимуму для того, чтобы упростить использование в средах, где строгая защита конфиденциальности может вызывать юридические проблемы.
4. Протоколы управления PKI должны позволять использование различных стандартизованных алгоритмов шифрования (в частности, RSA, DSA, MD5, SHA-1). Это значит, что любой CA, RA или конечный элемент в принципе может выбирать алгоритмы для своих ключевых пар.
5. Протоколам управления PKI недопустимо препятствовать генерации ключевых пар конечными элементами, вовлечёнными в процесс, а также агентами RA или CA. Генерация ключей может выполняться в разных местах, но для целей управления PKI можно считать, что генерация выполняется там, где ключ впервые появляется на конечном элементе, RA или CA.
6. Протоколы управления PKI должны поддерживать публикацию сертификатов вовлечёнными в процесс конечными элементами, а также агентами RA или CA. В разных реализациях и средах могут выбираться любые из перечисленных вариантов.
7. Протоколы управления PKI должны поддерживать создание списков отозванных сертификатов (CRL[1]), позволяя конечным элементам делать запросы на отзыв сертификатов. Это должно осуществляться таким способом, который не позволял бы простыми средствами организовать атаку на отказ служб (DoS[2]).
8. Протоколы управления PKI должны обеспечивать возможность работы на основе различных «транспортных» механизмов, включая, в частности, электронную почту http, TCP/IP и ftp.
9. Окончательное решение о создании сертификата остаётся за агентом CA. Ни RA, ни конечный элемент не могут предполагать, что выпущенный агентом CA сертификат будет в точности содержать запрошенные значения - CA может менять значения полей сертификата, добавлять, удалять или менять расширения в соответствии с правилами работы. Иными словами, все элементы PKI (конечные элементы, RA и CA) должны быть способны обрабатывать отклики на запросы сертификатов, в которых выдаваемые сертификаты отличаются от запрашиваемых (например, агент CA может сократить срок действия сертификата). Отметим, что правила могут запрещать агенту CA публиковать или иным способом распространять сертификат, пока запросившая сторона не рассмотрит и не примет созданный сертификат (обычно с помощью сообщения certConf).
10. Должен поддерживаться аккуратный, планируемый переход от одной пары ключей нескомпрометированного[3] агента CA к следующей паре (обновление ключа CA). Конечный элемент, среда PSE которого содержит новый открытый ключ CA (в результате обновления ключа CA), должен также быть способен проверить верифицируемость сертификатов с помощью старого открытого ключа. Конечные элементы, которые доверяют старой паре ключей CA непосредственно, должны быть способны также проверить сертификаты, подписанные с использованием нового секретного ключа CA (это требуется в ситуациях, когда старый открытый ключ CA являлся «аппаратной» частью криптографического оборудования элемента).
11. В некоторых реализациях и средах функции агента регистрации RA могут выполняться непосредственно агентом CA. Протоколы должны быть устроены так, чтобы конечные элементы взаимодействовали с агентами RA и CA в таких ситуациях по одному протоколу. Естественно, что конечный элемент для защиты при обмене данными должен использовать в таких случаях корректный открытый ключ RA для агента CA.
12. Когда конечный элемент запрашивает сертификат, содержащий заданное значение открытого ключа, этот конечный элемент должен быть готов продемонстрировать владение соответствующим закрытым ключом. Это можно реализовать разными способами в зависимости от типа запроса сертификата. В параграфе 4.3 описаны методы обмена данными по основному каналу (in-band), определённые для сообщений PKIX-CMP[4].

3.1.3. Операции управления PKI

На приведённом ниже рисунке показаны связи между определёнными ранее элементами в терминах операций управления PKI. Буквы на рисунке указывают «протоколы» в том смысле, что сообщения, определённые для управления PKI, могут передаваться по отмеченным на рисунке буквами линиям.

Набор операций верхнего уровня, для которых определены управляющие сообщения, показан ниже.

1. Организация CA. При организации нового агента CA некоторые этапы (например. Создание начальных CRL, экспорт открытого ключа CA) являются обязательными.
2. Инициализация конечного элемента включает импорт открытого ключа корневого CA и запрос информации об опциях, поддерживаемых элементом управления PKI.
3. Сертификация. Различные операции, приводящие к созданию новых сертификатов.
    1. *Начальная регистрация/сертификация.* Это процесс, в котором конечный элемент впервые представляет себя агенту CA или RA до эмиссии агентом CA сертификата или сертификатов для данного конечного элемента. Результатом этого процесса (при успешном завершении) является эмиссия агентом CA

[1]Certificate Revocation List.
[2]Denial-of-service attack.
[3]Отметим, что при компрометации ключа CA для всех элементов домена данного агента CA должна быть выполнена повторная инициализация.
[4]Certificate Management Protocol - протокол управления сертификатами.

открытого ключа для конечного элемента, возврат сертификата и/или отправка этого сертификата в публичное хранилище (репозиторий). Процесс может (и, как правило, будет) состоять из множества «этапов», которые могут включать инициализацию оборудования конечного элемента. Например, оборудование конечного элемента должно быть безопасно инициализировано с помощью открытого ключа CA, который будет использоваться при проверке путей сертификатов. Кроме того, конечные элементы обычно требуется инициализировать с использованием их собственных ключевых пар.

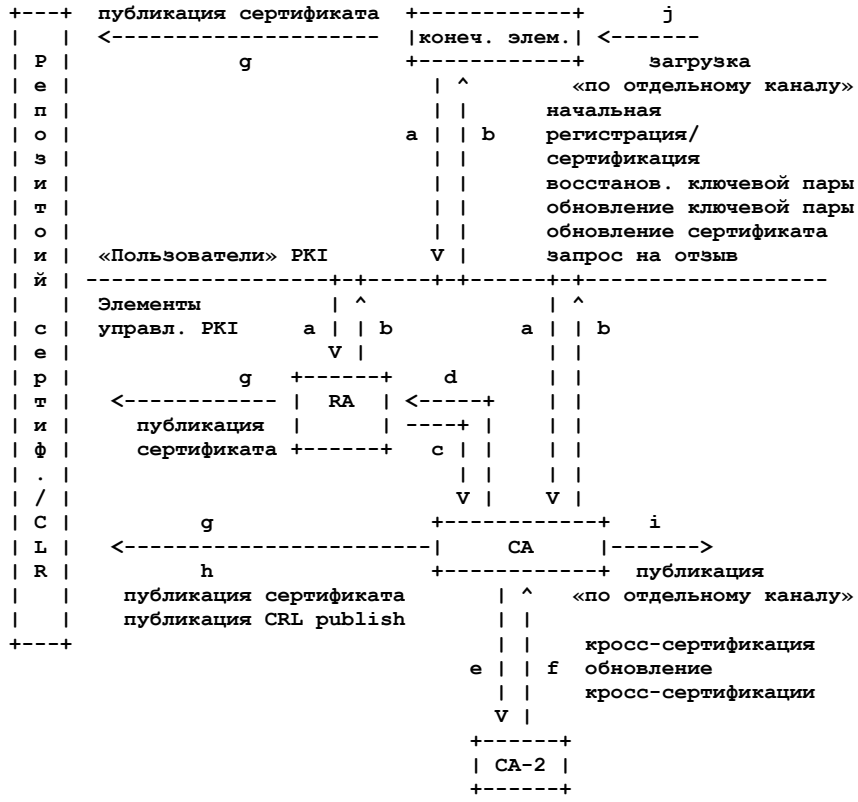

Рисунок 1. Элементы PKI.

2. *Обновление ключевой пары.* Каждую пару ключей требуется регулярно обновлять (т. е., заменять новой парой ключей) и выпускать новый сертификат.

3. *Обновление сертификата.* При завершении срока действия сертификата он может быть «обновлён», если в среде не произошло связанных с этим сертификатом изменений.

4. *Обновление пары ключей CA.* Как и для конечных элементов ключевые пары агентов CA требуется регулярно менять. Однако при такой замене требуются иные механизмы.

5. *Запрос кросс-сертификации.* Агент CA запрашивает эмиссию кросс-сертификата у другого CA. В данном стандарте «кросс-сертификатом» называется сертификат, в котором CA-субъект и CA-эмитент различаются, а поле SubjectPublicKeyInfo содержит ключ верификации (т. е., сертификат, выданный для пары ключей, используемой CA-субъектом в подписях). В некоторых случаях, когда требуется более тонкое различие, кросс-сертификат называется «междоменным» (участвующие в процессе агенты CA относятся к разным административным доменам) или «внутридоменным» (в остальных случаях).

   1. Примечание 1. Приведённое выше определение «кросс-сертификата» соответствует определению «сертификата CA» в стандарте X.509. Не следует путать этот термин с типом атрибута «cACertificate» в стандарте X.500.

   2. Примечание 2. Во многих средах термин «кросс-сертификат» трактуется, как синоним термина «междоменный кросс-сертификат», если явно не указано иное.

   3. Примечание 3. Эмиссия кросс-сертификатов может (но не обязана) быть взаимной, когда оба агента CA выпускают кросс-сертификаты один для другого.

6. *Обновление кросс-сертификата.* Похоже на обновление сертификата, но включает кросс-сертификацию.

4. Операции обнаружения сертификата/CRL. Некоторые операции управления PKI, приводящие к публикации сертификатов или CRL:

   1. *Публикация сертификата.* После решения проблем, связанных с созданием сертификата, требуются некоторые меры по его публикации. «Меры», определённые в PKIX **могут** включать сообщения, определённые в параграфах 5.3.13 - 5.3.16, или иные методы (например, LDAP), описанные в [RFC2559] и [RFC2585] (документы «Operational Protocols» в серии спецификаций PKIX).

   2. *Публикация CRL.* Аналогичная публикации сертификата.

5. Операции восстановления. Некоторые операции управления PKI, используемые при «потере» конечным элементом своей среды PSE:

   1. *Восстановление ключевой пары.* Опционально пользовательский ключевой материал (например, секретный ключ пользователя, используемый для расшифровки) **могут** сохраняться (резервная копия) агентами CA и RA или системой резервного копирования ключей, связанной с CA или RA. Если клиенту требуется восстановить сохранённый в резервной копии ключевой материал (например, в случае забытого

пароля или утраченного файла цепочки ключей[1]), для такого восстановления может потребоваться обмен протокольными сообщениями.

6. Операции отзыва. Операции управления PKI. приводящие к созданию новых записей CRL и/или новых CRL:
   1. *Запрос на отзыв.* Уполномоченное лицо может уведомить CA об аномальной ситуации, требующей отзыва сертификата.
7. Операции PSE. Хотя определение операций PSE (например, перемещение PSE, смена PIN и т. п.) выходит за пределы данной спецификации, мы определяем сообщение PKIMessage (CertRepMessage) которое может стать основой для таких операций.

Отметим, что интерактивные протоколы не являются единственным способом выполнения перечисленных выше операций. Для всех операций возможны автономные (off-line) методы достижения того же результата и данная спецификация не требует использования протоколов on-line. Например, при использовании аппаратных маркеров, многие операции **могут** быть выполнены в процессе физической доставки маркеров.

В последующих разделах определяется набор стандартных сообщений, обеспечивающих выполнение перечисленных выше операций. Транспортные протоколы для передачи этих сообщений в разных средах (обмен файлами, интерактивное взаимодействие, электронная почта и WWW) выходят за пределы данной спецификации и рассматриваются в отдельных документах.

4. Допущения и ограничения

4.1. Инициализация конечного элемента

Первым шагом конечного элемента при работе с объектами управления PKI является запрос информации о поддерживаемых функциях PKI и защищённое получение копии соответствующего открытого ключа (или ключей) корневого агента CA.

4.2. Начальная регистрация/сертификация

Существует множество схем, которые могут применяться для начальной регистрации и сертификации конечных элементов. Ни один из методов не подходит «на все случаи жизни» по причине широкого спектра правил, которые могут быть реализованы в CA, и множества вариантов конечных элементов.

Однако можно классифицировать схемы начальной регистрации/сертификации, которые поддерживаются данной спецификацией. Отметим, что слово «начальная» в предыдущем предложении играет важную роль — мы имеем дело с ситуацией, когда рассматриваемый конечный элемент ранее не имел контактов с PKI. Когда конечный элемент уже владеет ключами сертификации, возможно некоторое упрощение/изменение процедуры.

Имея классифицированные схемы, поддерживаемые данной спецификацией, мы можем задать некоторые из этих схем в качестве обязательных, а некоторые - в качестве опциональных. Цель состоит в том, чтобы обязательные схемы перекрывали достаточное число случаев, возникающих на практике, а для менее распространённых ситуаций имелись дополнительные схемы. Таким способом достигается компромисс между уровнем гибкости и простотой реализации.

Далее приводится классификация схем начальной регистрации/сертификации.

4.2.1. Используемые критерии

4.2.1.1. Инициирование регистрации/сертификации

В терминах создаваемых сообщений PKI можно рассматривать инициирование обмена при начальной регистрации/сертификации, как создание первого сообщения PKI, связанного с этим конечным элементом. Отметим, что в реальности процедура начальной регистрации/сертификации может происходить иначе (например, отдел персонала может обратиться по телефону к оператору RA).

К числу возможных мест относятся конечный элемент, RA или CA.

4.2.1.2. Идентификация источника сообщения от конечного элемента

Интерактивные (on-line) сообщение, создаваемые конечным элементом, которому нужен сертификат, могут быть или не быть идентифицированными. Требование заключается в проверке подлинности источника любых сообщений от конечного элемента к PKI (CA/RA).

В данной спецификации такая проверка осуществляется PKI (CA/RA), вводящим конечный элемент с секретным значением (начальный ключ аутентификации) и эталонным значением (служит для идентификации секретного значения) с использованием неких иных каналов передачи информации (out-of-band). После этого можно использовать начальный ключ аутентификации для защиты соответствующих сообщений PKI.

Таким образом, можно классифицировать схемы начальной регистрации/сертификации по наличию проверки подлинности интерактивных сообщений от конечного элемента к PKI.

Примечание 1. Здесь не рассматривается аутентификация сообщений от PKI к конечному элементу, поскольку такая аутентификация о**бязательна**. В любом случае эта аутентификация может быть обеспечена после получения открытого ключа корневого CA оборудованием конечного элемента или выполнена с использованием ключа начальной аутентификации.

Примечание 2. Процедура начальной регистрации/сертификации может быть защищена за счёт аутентификации сообщений от конечного элемента с использованием тех или иных «автономных» (out-of-band) мер (например, передача «из рук в руки»).

4.2.1.3. Место генерации ключа

[1]Key chain file.

В данной спецификации говорят о месте «генерации ключа», как о точке, где открытый или закрытый ключ пары впервые появляется в сообщении PKIMessage. Отметим, что это не исключает использования услуг централизованной генерации ключей - на практике пара ключей **может** генерироваться в любом месте и доставляться конечному элементу RA или CA с использованием (фирменного или стандартизованного) протокола запросов/откликов для генерации ключей[1].

Таким образом, местом генерации ключа может являться конечный элемент, RA или CA.

4.2.1.4. Подтверждение успешной сертификации

После создания начального сертификата для конечного элемента могут быть обеспечены дополнительные гарантии за счёт передачи конечным элементом явного подтверждения успешного получения (или индикации создания0 сертификата. Естественно, такое подтверждение должно быть защищено (с использованием ключа начальной аутентификации или иным способом).

Возможны два варианта - использовать или не использовать подтверждения.

4.2.2. Обязательные схемы

Приведённые выше критерии допускают существование множества схем регистрации/сертификации. Данная спецификация **требует** от оборудования конечных элементов, CA и RA **обязательной** поддержки второй из рассмотренных ниже схем (параграф 4.2.2.2). Любой элемент **может** также дополнительно поддерживать иные схемы.

4.2.2.1. Централизованная схема

В терминах приведённой выше классификации эта схема в некоторых случаях является простейшей из возможных:

- инициирование выполняется на сертифицирующем CA;
- не требуется интерактивных сообщений для аутентификации;
- «генерация ключей» происходит на сертифицирующем CA (см. параграф 4.2.1.3);
- не требуется подтверждающих сообщений.

В терминах потока сообщений эта схема означает, что требуется лишь одно сообщение, которое передаётся конечному элементу от CA. Это сообщение должно содержать всю среду PSE для конечного элемента. Должны также использоваться некоторые автономные способы обеспечения конечному элементу возможности аутентификации полученного сообщения и расшифровки любых зашифрованных значений.

4.2.2.2. Базовая схема аутентификации

В терминах приведённой выше классификации эта схема включает:

- инициирование, выполняемое на стороне конечного элемента;
- **требуется** аутентификация сообщения;
- «генерация ключей» происходит на стороне конечного элемента (см. параграф 4.2.1.3);
- **требуется** подтверждающее сообщение.

В терминах потока сообщений базовая схема имеет вид:

Конечный элемент		***RA/CA***
«Автономное» распространение начального ключа аутентификации (IAK[2]) и эталонного значения (RA/CA -> EE)		
Генерация ключа		
Запрос на создание сертификата		
Защита запроса с помощью IAK		
	-->>-- запрос сертификации -->>--	
		Проверка запроса
		Обработка запроса
		Создание отклика
	--<<-- сертификационный отклик --<<--	
Обработка отклика		
Создание подтверждения		
	-->>-- сообщение cert conf -->>--	
		Проверка подтверждения
		Создание отклика
	--<<-- conf ack (необязательно) --<<--	
Обработка отклика		

(В тех случаях, когда верификация с использованием подтверждения сертификата не удаётся, агент RA/CA **должен** отозвать этот сертификат, если он был опубликован или иным путём сделан доступным).

4.3. Подтверждение владения секретным ключом

Для предотвращения некоторых типов атак и обеспечения агентам CA/RA возможности проверить корректность связки между конечным элементом и парой ключей здесь заданы операции управления PKI, которые позволяют конечной точке подтвердить, что её владение секретным ключом (т. е., возможность использования ключа) соответствует открытому ключу, для которого запрашивается сертификат. Агент CA/RA свободен в выборе способа реализации POP[3] (например, автономные пути или сообщения PKIX-CMP по основному каналу) в своём сертификационном обмене

[1] Рассмотрение таких протоколов выходит за пределы данной спецификации.
[2] Initial Authentication Key.
[3] Proof-of-Possession.

(например, выбор может определяться принятой политикой защиты). Однако от агентов CA/RA **требуется** выполнение POP тем или иным способом, поскольку в настоящее время используется множество протоколов, не относящихся к PKIX (например, протоколы обмена электронной почтой), которые не проверяют в явной форме привязку секретного ключа к конечному элементу. Пока повсеместно не используется протокол, выполняющий проверку привязки (для ключевых пар подписи, шифрования и согласования ключей), приходиться полагаться на то, что такая привязка проверяется агентом CA/RA. Следовательно, если привязка не проверяется агентом CA/RA, сертификаты в инфраструктуре Internet PKI, по сути, утрачивают своё значение.

POP обеспечивается разными способами в зависимости от типа ключа, для которого запрашивается сертификат. Если ключ может использоваться с разными целями (например, ключ RSA), **может** применяться любой подходящий метод (например, ключ, который может служить для подписи, а также для других целей, **не следует** отправлять агенту CA/RA с целью подтверждения владения).

Эта спецификация явно разрешает случаи, когда конечная точка представляет приемлемое подтверждение агенту RA, а тот сообщает агенту CA, что требуемые подтверждения получены (и проверены!). Например, конечный элемент, желающий хранить сертифицированный ключ подписи, может отправить подходящую подпись агенту RA, который просто уведомит соответствующий CA о том, что конечный элемент предоставил требуемое подтверждение. Естественно, что некоторые варианты политики могут запрещать такой подход (например, проверка POP в процессе сертификации может быть разрешена только агентам CA).

4.3.1. Ключи подписи

Применительно к ключам подписи конечный элемент может подписать значение для подтверждения владения секретным ключом.

4.3.2. Ключи шифрования

В случае ключей шифрования конечный элемент может предоставлять агенту CA/RA секретный ключ или расшифровывать данные в качестве подтверждения обладания секретным ключом (см. параграф 5.2.8). Расшифровка данных может осуществляться с использованием прямого или косвенного метода.

Метод прямой расшифровки требует от агента RA/CA ввода случайного запроса, на который конечный элемент должен ответить незамедлительно.

При использовании косвенной расшифровки вводится сертификат, который шифруется для конечного элемента (и позволяет тому продемонстрировать свою способность расшифровать этот сертификат сообщение и передать его в подтверждении). Это позволяет агенту CA вводить сертификат в той форме, которая может быть использована только соответствующим конечным элементом.

Данная спецификация рекомендует использовать косвенный метод, поскольку он не требует передачи дополнительных сообщений (подтверждение владения может быть обеспечено с использованием трёх обычных сообщений - {запрос, отклик, подтверждение}.

4.3.3. Ключи согласования ключей

Для ключей согласования ключей конечный элемент и элемент управления PKI (т. е., CA или RA) должны организовать разделяемый секретный ключ для подтверждения того, что конечный элемент владеет секретным ключом.

Отметим, что это не вносит каких-либо ограничений на ключи, которые могут быть сертифицированы данным CA. В частности, для ключей Diffie-Hellman конечный элемент может свободно выбирать параметры алгоритма, обеспечивающие агенту CA возможность генерации краткосрочных (или одноразовых) ключевых пар с подходящими параметрами, когда это требуется.

4.4. Обновление ключа корневого CA

Это обсуждение применимо только к агентам CA, которые непосредственно являются доверенными для некоторых конечных элементов. Самоподписанные CA **следует** рассматривать, как непосредственно доверенные агенты CA. Решение вопроса о признании несамоподписанного агента CA в качестве непосредственно доверенного для некоторых конечных элементов определяется политикой CA и, следовательно, выходит за пределы данной спецификации.

Описанная здесь процедура основана на том, что CA защищает свой новый открытый ключ с помощью предыдущего секретного ключа и наоборот. Таким образом, при обновлении агентом CA пары ключей он должен генерировать два избыточных атрибута cACertificate, если сертификаты делаются доступными с использованием каталогов X.500 (всего генерируется 4 атрибута — OldWithOld, OldWithNew, NewWithOld и NewWithNew).

Когда CA меняет ключевую пару, те элементы, которые обладают старым открытым ключом CA, полученным по автономным каналам (out-of-band), сильнее «страдают» от такой замены. Это будут те конечные элементы, которым требуется доступ к новому открытому ключу CA, защищённому старым секретным ключом CA. Однако эта сложность будет возникать только на ограниченный период (пока конечному элементу не будет доставлен новый открытый ключ CA с использованием «автономного» механизма). Обычно это происходит при истечении срока действия сертификатов данного конечного элемента.

Структурой данных, используемой для защиты старого и нового открытых ключей CA, является стандартный сертификат (который может включать расширения). Новых структур данных не требуется.

Примечание 1. В этой схеме не используется расширений X.509 v3, поэтому она позволяет работать даже с сертификатами версии 1. Использование расширения KeyIdentifier будет повышать эффективность.

Примечание 2. Хотя эта схема может быть обобщена для случаев, когда CA обновляет свою ключевую пару более одного раза в течение срока действия сертификатов конечных элементов, ценность такого обобщения представляется сомнительной. Отказ от такого обобщения просто означает, что сроки действия сертификатов, эмиттированных со старой парой ключей CA, не могут превышать срока окончания действия OldWithNew.

Примечание 3. Эта схема гарантирует, что конечные элементы будут получать (по «автономным» каналам) новый открытый ключ CA не позднее завершения срока действия имеющегося у конечного элемента сертификата, подписанного старым секретным ключом CA. В других случаях операции обновления ключей и/или сертификата не обязательно будут требовать доставки ключей по «автономным» каналам (это зависит от оборудования конечного элемента).

4.4.1. Действия оператора CA

Для смены ключа CA оператор CA должен выполнить перечисленные ниже операции.

1. Генерация новой пары ключей;
2. создание сертификата со старым открытым ключом CA, подписанного новым секретным ключом (сертификат «старый с новым»);
3. создание сертификата с новым открытым ключом CA, подписанного старым секретным ключом (сертификат «новый со старым»);
4. создание сертификата с новым открытым ключом CA, подписанного новым секретным ключом (сертификат «новый с новым»);
5. публикация созданных сертификатов через репозиторий и/или иными способами (например, с использованием сообщений CAKeyUpdAnn);
6. экспорт нового открытого ключа CA, позволяющий конечным элементам получать этот ключ с использованием «автономных» механизмов (если это нужно).

Старый секретный ключ CA больше не требуется, однако старый открытый ключ CA будет использоваться ещё некоторое время. Потребность в старом открытом ключе CA исчерпывается (не в результате отказа от него), когда все конечные элемента этого агента CA получат защищённым путём новый открытый ключ CA.

Сертификат «старый с новым» должен иметь срок действия с момента генерации старой пары ключей до момента завершения срока действия старого открытого ключа.

Сертификат «новый со старым» должен иметь срок действия с момента генерации новой пары ключей до момента завершения процессов получения нового открытого ключа CA всеми конечными элементами этого агента CA (но не позднее момента завершения срока действия старого открытого ключа).

Сертификат «новый с новым» должен иметь срок действия с момента генерации новой пары ключей до момента, предшествующего следующему обновлению ключевой пары агента CA.

4.4.2. Проверка сертификатов

Обычно при верификации подписи проверяющая сторона (наряду с другим) смотрит сертификат, содержащий открытый ключ подписывающей стороны. Однако, поскольку CA разрешено обновлять свои ключи, возникает несколько случаев, перечисленных в таблице.

	Репозиторий содержит **новый** и **старый** открытый ключ		Репозиторий содержит только **старый** открытый ключ (например, в результате задержки публикации)	
	PSE содержит **новый** открытый ключ	PSE содержит **старый** открытый ключ	PSE содержит **новый** открытый ключ	PSE содержит **старый** открытый ключ
Сертификат подписывающей стороны защищён **новым** открытым ключом	Случай 1 Стандартная ситуация, когда проверяющая сторона может напрямую верифицировать сертификат без использования репозитория	Случай 3 В этом случае проверяющая сторона должна обратиться к репозиторию для получения **нового** открытого ключа	Случай 5 Хотя оператор CA не обновил репозиторий, проверяющая сторона может верифицировать сертификат напрямую (как в случае 1)	Случай 7 Оператор CA не обновил репозиторий и проверка приведёт к **отказу**
Сертификат подписывающей стороны защищён **старым** открытым ключом	Случай 2 В этом случае проверяющая сторона должна обратиться к репозиторию для получения **старого** открытого ключа	Случай 4 В этом случае проверяющая сторона может напрямую верифицировать сертификат без использования репозитория	Случай 6 Проверяющая сторона будет воспринимать эту ситуацию, как случай 2, и обращаться к репозиторию; верификация приведёт в **отказу**	Случай 8 Хотя оператор CA не обновил репозиторий, проверяющая сторона может верифицировать сертификат напрямую (как в случае 4)

4.4.2.1. Верификация для случаев 1, 4, 5, 8

В этих случаях проверяющая сторона имеет локальную копию открытого ключа CA, которая может напрямую использоваться для верификации открытого ключа. Этот случай не отличается от ситуации, когда никаких изменений не происходило.

Отметим, что случай 8 может наблюдаться с интервале между генерацией агентом CA новой пары ключей и размещением оператором CA новых атрибутов в репозитории. Случай 5 может возникать, если оператор CA эмиттировал сертификаты подписывающей и верифицирующей стороны в течение указанного выше интервала (оператору CA **следует** избегать таких ситуаций, поскольку они могут приводить к отказам, как отмечено ниже).

4.4.2.2. Верификация для случая 2

В случае 2 проверяющая сторона должна иметь старый открытый ключ CA. Порядок действий проверяющего:

1. просмотр атрибута caCertificate в репозитории и получение сертификата OldWithNew (определяется на основе срока действия; отметим, что поля субъекта и эмитента должны совпадать);
2. проверка корректности с использованием нового ключа CA (имеется на проверяющей стороне);
3. при позитивном результате проверки проверяется сертификат подписавшей стороны с использованием старого ключа CA.

Случай 2 возникает, когда оператор CA эмиттировал сертификат подписавшей стороны, потом сменил ключ и после этого эмиттировал сертификат проверяющей стороны - такая ситуация вполне типична.

4.4.2.3. Верификация для случая 3

В случае 3 проверяющая сторона должна иметь доступ к новому открытому ключу CA. Порядок действий проверяющего:

1. просмотр атрибута caCertificate в репозитории и получение сертификата NewWithOld (определяется на основе срока действия; отметим, что поля субъекта и эмитента должны совпадать);
2. проверка корректности с использованием старого ключа CA (имеется на проверяющей стороне);
3. при позитивном результате проверки проверяется сертификат подписавшей стороны с использованием нового ключа CA.

Случай 2 возникает, когда оператор CA эмиттировал сертификат проверяющей стороны, потом сменил ключ и после этого эмиттировал сертификат подписавшей стороны - такая ситуация вполне типична.

4.4.2.4. Отказ при верификации для случая 6

В этом случае CA эмиттирует среду PSE проверяющей стороны, которая содержит новый ключ без обновления атрибутов в репозитории. Это означает, что проверяющий не может надёжным путём получить старую версию ключа CA и верификация завершается отказом.

Отметим, что отказ возникает по вине оператора CA.

4.4.2.5. Отказ при верификации для случая 7

В этом случае CA эмиттирует сертификат подписывающей стороны, защищённый новым ключом, без обновления атрибутов в репозитории. Это означает, что проверяющий не может надёжным путём получить новую версию ключа CA и верификация завершается отказом.

Отметим, что отказ возникает по вине оператора CA.

4.4.3. Отзыв - смена ключа CA

Как было сказано выше, верификация сертификата усложняется, если агенту CA разрешено менять свои ключи. Это верно также для случаев проверки при отзыве, поскольку агент CA может иметь подпись CRL, выполненную с использованием более нового ключа, нежели для пользовательской среды PSE.

Анализ вариантов для отзыва совпадает с анализом вариантов при проверке сертификата.

5. Структуры данных

В этом разделе описаны структуры данных, требуемые для управляющих сообщений PKI. В разделе 6 описаны ограничения для значений и последовательности событий для каждой из операций управления PKI.

5.1. Сообщение PKI в целом

Все сообщений, используемые в данной спецификации для целей управления PKI имеют структуру, показанную ниже.

```
 PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL
}
PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage
```

Поле PKIHeader для многих типов сообщений PKI содержит однотипную информацию.

Поле PKIBody содержит специфическую для данного сообщения информацию.

Если используется поле PKIProtection, оно содержит биты, служащие для защиты сообщения PKI.

Поле extraCerts может содержать сертификаты, которые могут быть полезны для получателя. Например, это поле может использоваться агентом CA или RA для информирования конечного элемента о сертификатах, которые тому потребуются для верификации нового сертификата агента (например, в случае, когда эмиттирующий сертификат конечного элемента агент CA не является коревым CA для данного конечного элемента). Отметим, что это поле не обязательно содержит путь к сертификату - получатель может сортировать это поле, делать из него выборку и т. п.

5.1.1. Заголовок сообщения PKI

Для всех сообщений PKI требуется некоторая информация в заголовке, служащая для адресации и идентификации транзакций. Часть такой информации будет присутствовать также в зависящем от транспорта «конверте». Однако, если сообщение PKI защищено, эта информация также будет защищена (т. е., мы не можем строить предположений о защищённом транспорте).

Для заголовков используется показанная ниже структура данных:

```
PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2) },
    sender              GeneralName,
    recipient           GeneralName,
    messageTime     [0] GeneralizedTime         OPTIONAL,
    protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    transactionID   [4] OCTET STRING            OPTIONAL,
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    freeText        [7] PKIFreeText             OPTIONAL,
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue     OPTIONAL
}
PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
```

Значение поля pvno = 2 соответствует для данной версии спецификации.

Поле sender содержит имя отправителя сообщения PKIMessage. Этого имени (вместе со значениям поля senderKID, если оно присутствует) должно быть достаточно для индикации ключа, который следует использовать для верификации защиты сообщения. Если передающая сторона ничего не знает об отправителе (например, начальное сообщение, когда конечный элемент ещё не знает своего отличительного имени DN[1], адреса электронной почты, адреса IP и т. д.), в поле sender **должно** помещаться значение NULL (т. е., SEQUENCE OF для отличительных имен имеет нулевой размер). В таких случаях поле senderKID **должно** содержать идентификатор (например, номер), показывающий получателю подходящий разделяемый секрет, который можно использовать для верификации этого сообщения.

Поле recipient содержит имя получателя PKIMessage. Это имя (вкупе со значением поля recipKID, если оно присутствует) должно обеспечивать возможность использования для верификации защиты сообщения.

Поле protectionAlg задаёт алгоритм, используемый для защиты сообщения. Если биты защиты не заданы (отметим, что защита PKIProtection является **опциональной**), это поле **должно** быть опущено. При наличии битов защиты это поле **должно** присутствовать.

Поля senderKID и recipKID служат для индикации ключей, которые используются для защиты сообщения (recipKID обычно требуется лишь в случаях использования для защиты ключей DH[2].

Эти поля **должны** при необходимости использоваться для уникальной идентификации ключа (например, при наличии нескольких ключей, связанных с данным отправителем), а в остальных случаях их **следует** опускать.

Поле transactionID в заголовке сообщения позволяет получателю сообщения связать это сообщение с транзакцией. Это требуется для всех транзакций, которые содержат что-либо сверх одной пары «запрос-отклик». Для транзакций, включающих только одну пару «запрос-отклик» клиент **может** заполнять поле transactionID в запросе. Если сервер получает запрос с установленным полем transactionID, он **должен** установить в поле transactionID передаваемого отклика то же значение. Если сервер получает запрос без поля transactionID, он **может** установить значение transactionID в отклике.

Для транзакций, которые включают что-либо сверх одной пары «запрос-отклик» клиенту **следует** генерировать transactionID для первого запроса. Если сервер получает запрос с установленным полем transactionID, он **должен** установить в поле transactionID передаваемого отклика то же значение. Если сервер получает запрос без поля transactionID, он **должен** заполнить в отклике поле transactionID самостоятельно созданным значением идентификатора. Во всех последующих запросах и откликах **должно** использоваться это значение transactionID. Во всех случаях использования transactionID для клиента **недопустимо** иметь более одной обрабатываемой в данный момент транзакции с одинаковым значением transactionID (для данного сервера). Серверы по своему усмотрению могут требовать (или не требовать) уникальности значений transactionID для корректной привязки сообщений к соответствующей транзакции. Обычно это означает, что сервер будет требовать уникальности пары {client, transactionID} или даже уникальности transactionID, если он не может различать клиентов на основе информации транспортного уровня. Сервер, получивший в транзакции, требующей более, чем одна пара сообщений «запрос-отклик», первое сообщение, которое содержит значение transactionID, не позволяющее выполнить приведённые выше требования (обычно в результате того, что такое значение transactionID уже используется) **должен** вернуть клиенту ErrorMsgContent с полем PKIFailureInfo, имеющим значение transactionIdInUse. Клиентам **рекомендуется** помещать в поле transactionID 128-битовое (псевдо)случайное значение, связанное с началом транзакции, чтобы снизить вероятность совпадения с уже используемым сервером значением transactionID.

Поля senderNonce и recipNonce служат для защиты сообщения PKIMessage от атак с повторным использованием3. Поле senderNonce обычно содержит 128 битов (псевдо) случайных данных, генерируемых отправителем, а поле recipNonce является копией поля senderNonce из предыдущего сообщения в данной транзакции.

Поле messageTime содержит временную метку момента генерации сообщения. Это позволяет конечным элементам установить/проверить свои часы для обеспечения синхронизации с центральной системой.

Поле freeText может служить для передачи понятных человеку сообщений (на любом языке или множестве языков). Язык, используемый в последовательности первым, является наиболее предпочтительным для откликов.

Поле generalInfo может служить для передачи получателю дополнительных данных, предназначенных для машинной обработки. Ниже описаны расширения generalInfo, которые определены в настоящее время и **могут** поддерживаться.

5.1.1.1. ImplicitConfirm

Используется EE для информирования агента CA о нежелании передавать подтверждения для эмиттированных сертификатов.

```
implicitConfirm OBJECT IDENTIFIER ::= {id-it 13}
```

[1]Distinguished Name.
[2]Diffie-Hellman.

```
        ImplicitConfirmValue ::= NULL
```

Если CA принимает запрос EE, он **должен** включить такое же расширение в заголовок PKIHeader своего отклика. Если EE не находит в отклике расширения, он **должен** передать подтверждение сертификата.

5.1.1.2. ConfirmWaitTime

Используется агентом CA для информирования EE о сроке ожидания подтверждения сертификата перед отзывом того и удалением транзакции.

```
        confirmWaitTime OBJECT IDENTIFIER ::= {id-it 14}
        ConfirmWaitTimeValue ::= GeneralizedTime
```

5.1.2. Тело сообщения PKI

```
PKIBody ::= CHOICE {
        ir       [0]  CertReqMessages,        --Запрос инициализации
        ip       [1]  CertRepMessage,         --Инициализационный отклик
        cr       [2]  CertReqMessages,        --Запрос сертификации
        cp       [3]  CertRepMessage,         --Сертификационный отклик
        p10cr    [4]  CertificationRequest,   --Запрос сертификата PKCS #10
        popdecc  [5]  POPODecKeyChallContent --Запрос pop
        popdecr  [6]  POPODecKeyRespContent, --Отклик pop
        kur      [7]  CertReqMessages,        --Запрос на обновление ключа
        kup      [8]  CertRepMessage,         --Отклик при обновлении ключа
        krr      [9]  CertReqMessages,        --Запрос на восстановление ключа
        krp      [10] KeyRecRepContent,       --Отклик при восстановлении ключа
        rr       [11] RevReqContent,          --Запрос на отзыв
        rp       [12] RevRepContent,          --Отклик при отзыве
        ccr      [13] CertReqMessages,        --Запрос кросс-сертификации
        ccp      [14] CertRepMessage,         --Отклик при кросс-сертификации
        ckuann   [15] CAKeyUpdAnnContent,     --Анонс обновления ключа CA
        cann     [16] CertAnnContent,         --Анонс сертификата
        rann     [17] RevAnnContent,          --Анонс отзыва
        crlann   [18] CRLAnnContent,          --Анонс CRL
        pkiconf  [19] PKIConfirmContent,      --Подтверждение
        nested   [20] NestedMessageContent,   --Вложенное сообщение
        genm     [21] GenMsgContent,          --Сообщение общего назначения
        genp     [22] GenRepContent,          --Отклик общего назначения
        error    [23] ErrorMsgContent,        --Сообщение об ошибке
        certConf [24] CertConfirmContent,     --Подтверждение сертификата
        pollReq  [25] PollReqContent,         --Запрос опроса
        pollRep  [26] PollRepContent          --Отклик при опросе
}
```

Эти типы описаны ниже в параграфе 5.3.

5.1.3. Защита сообщений PKI

Для некоторых сообщений PKI применяется защита целостности (отметим, что при защите с помощью асимметричного алгоритма источник сообщения может быть засвидетельствован автоматически, если соответствующая открытая компонента уже сертифицирована; если открытая компонента не сертифицирована, источник не может засвидетельствован автоматически, но это может быть сделано с использованием «автономных» способов).

Для защиты используется структура

```
PKIProtection ::= BIT STRING
```

Входной информацией для расчёта PKIProtection является DER-представление структуры данных

```
ProtectedPart ::= SEQUENCE {
        header    PKIHeader,
        body      PKIBody
}
```

Могут возникать ситуации, когда строка битов PKIProtection осознанно не используется в целях защиты сообщения (т. е., это необязательное поле опускается), поскольку для защиты сообщения будут применяться иные средства (внешние по отношению к PKIX). Данная спецификация явно разрешает такое поведение. Примеры внешней защиты включают инкапсуляцию PKCS #7 [PKCS7] и Security Multiparts [RFC1847] для сообщений PKIMessage (или просто PKIBody без тега CHOICE, если для данных заголовка PKIHeader обеспечивается внешний механизм защиты). Отметим, однако, что для большинства таких внешних механизмов требуется наличие у конечной точки сертификата открытого ключа и/или уникального отличительного имени DN, и/или другой информации, связанной с инфраструктурой. В результате такой подход может оказаться неприемлемым для начальной регистрации, восстановления ключей или других процессов с характеристиками «самозагрузки». Для таких случаев может оказаться необходимым использование параметра PKIProtection. В будущем, когда/если внешние механизмы изменятся и смогут поддерживать самозагрузку, использование PKIProtection может стать редким или прекратиться совсем.

В зависимости от ситуации биты PKIProtection могут содержать код идентификации сообщения (MAC[1]) или цифровую подпись. Ниже перечислены все возможные варианты.

5.1.3.1. Разделяемый секрет

В этом случае отправитель и получатель совместно владеют некой секретной информацией (для получения секрета используются автономные механизмы или предшествующие операции управления PKI). Поле PKIProtection будет содержать значение MAC, а protectionAlg будет иметь вид (см. также Приложение D.2):

```
id-PasswordBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
        salt                OCTET STRING,
```

[1]Message Authentication Code.

```
        owf                 AlgorithmIdentifier,
        iterationCount      INTEGER,
        mac                 AlgorithmIdentifier
}
```

В приведённом выше protectionAlg значение поля salt (затравка) добавляется в конце ввода разделяемого секрета. После этого алгоритм OWF применяется iterationCount раз, секретное значение с добавленной в конце затравкой служит входными данными для первой итерации, а результат каждой итерации используется в качестве входных данных для следующей итерации. Результат последней итерации (для краткости его называют BASEKEY), размером "H" используется для формирования симметричного ключа. Если алгоритм MAC требует ключа размером K битов и K <= H, будут использоваться K старших битов BASEKEY. Если K > H, все биты BASEKEY используются в качестве H старших битов ключа, OWF("1" || BASEKEY) образуют следующую группу из H битов ключа, OWF("2" || BASEKEY) — третью группу и т. д., пока не будет достигнут размер K ("N" означает ASCII-представление числа N, а "||" - конкатенацию).

Примечание. **Рекомендуется** сохранять значения полей параметра PBMParameter неизменными на протяжении одной транзакции (например, ir/ip/certConf/pkiConf) для снижения издержек, связанных с вычислением PasswordBasedMac.

5.1.3.2. Пары ключей DH

Когда отправитель и получатель обладают сертификатами Diffie-Hellman с совместимыми параметрами DH, для защиты сообщения конечный элемент должен генерировать симметричный ключ на основе значений своего секретного ключа DH и открытого DH-ключа получателя сообщения PKI. Поле PKIProtection будет содержать значение MAC, полученное с помощью этого симметричного ключа, а protectionAlg будет иметь вид:

```
id-DHBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 30}

DHBMParameter ::= SEQUENCE {
        owf                 AlgorithmIdentifier,
        -- Идентификатор алгоритма для необратимой функции (рекомендуется SHA-1)
        mac                 AlgorithmIdentifier
        -- Идентификатор алгоритма MAC (например, DES-MAC, Triple-DES-MAC [PKCS11],
}       -- или HMAC [RFC2104, RFC2202])
```

В приведённом выше protectionAlg алгоритм OWF применяется к результату расчёта по методу Diffie-Hellman. результат OWF (для краткости называется BASEKEY) размером "H" используется для формирования симметричного ключа. Если алгоритм MAC требует ключа размером K битов и K <= H, будут использоваться K старших битов BASEKEY. Если K > H, все биты BASEKEY используются в качестве H старших битов ключа, OWF("1" || BASEKEY) образуют следующую группу из H битов ключа, OWF("2" || BASEKEY) — третью группу и т. д., пока не будет достигнут размер K ("N" означает ASCII-представление числа N, а "||" - конкатенацию).

5.1.3.3. Подпись

В этом случае отправитель владеет парой ключей цифровой подписи и просто подписывает сообщение PKI. Поле PKIProtection будет содержать значение подписи, а protectionAlg будет содержать идентификатор алгоритма AlgorithmIdentifier для цифровой подписи (например, md5WithRSAEncryption или dsaWithSha-1).

5.1.3.4. Комплексная защита

В случаях, когда конечный элемент передаёт защищённое сообщение PKI агенту RA, последний **может** переслать это сообщение агенту CA, добавив к сообщению свою защиту (которая **может** представлять собой MAC или подпись в зависимости от сертификатов, согласованных между RA и CA). Это осуществляется путём вложения сообщения от конечного элемента целиком в новое сообщение PKI. Используемая для этого структура показана ниже.

```
        NestedMessageContent ::= PKIMessages
```

(использование сообщений PKIMessage в форме SEQUENCE OF PKIMessage позволяет агенту RA обрабатывать в пакетном режиме запросы нескольких EE в форме одного нового сообщения; для простоты все сообщения такого пакета **должны** быть однотипными - например, ir). Если RA хочет изменить сообщение (сообщения) тем или иным способом (например, добавить новые значения полей или расширения), он **может** создать свою желаемую структуру PKIBody. Исходное сообщение PKIMessage от EE **может** быть включено в поле generalInfo заголовка PKIHeader (например, для случаев, когда CA пожелает проверить POP или другие данные из исходного сообщения). Поле infoType, используемое в таких ситуациях имеет форму {id-it 15} (см. параграф 5.3.19, где описаны значения id-it), а infoValue представляет собой сообщения PKIMessage (содержимое **должно** быть представлено в том же порядке, как располагаются запросы в PKIBody).

5.2. Структуры данных общего назначения

Перед описанием конкретных типов, которые могут быть включены в тело сообщения PKIBody, определим используемые неоднократно структуры данных.

5.2.1. Содержимое запрашиваемого сертификата

Различные управляющие сообщения PKI требуют, чтобы инициатор сообщения указывал некоторые поля, присутствие которых в сертификате обязательно. Структура данных CertTemplate позволяет конечному элементу или агенту RA задавать свои требования к сертификату. Структура CertTemplate идентична структуре Certificate, но некоторые поля не являются обязательными.

Отметим, что даже при полном задании инициатором требований к сертификату, агент CA может менять поля в эмиттируемом сертификате. Если изменённый сертификат неприемлем для запрашивающей стороны, та **должна** вернуть сообщение certConf, не включающее данный сертификат (через CertHash) или включающее его (через CertHash) со статусом rejected. Определение и описание применения CertHash и certConf дано в параграфе 5.3.18.

См. Приложение C и [CRMF], где описан синтаксис CertTemplate.

5.2.2. Зашифрованные значения

При передаче зашифрованных значений (данная спецификация ограничивает шифрование использованием секретных ключей или сертификатов) в сообщениях PKI используется структура данных EncryptedValue.

Синтаксис EncryptedValue описан в [CRMF].

Использование этой структуры данных требует от создателя и предполагаемого получателя возможности зашифровки и расшифровки, соответственно. Обычно это означает, что отправитель и получатель имеют или могут генерировать разделяемый секретный ключ.

Если получатель сообщения PKIMessage уже владеет секретным ключом, используемым для дешифровки, поле encSymmKey **может** содержать сеансовый ключ, зашифрованный с использованием открытого ключа получателя.

5.2.3. Коды состояний и информация об отказах для сообщений PKI

Все сообщения-отклики будут включать ту или иную информацию о состоянии. Ниже перечислены коды состояний.

```
PKIStatus ::= INTEGER {
    accepted                (0), -- воспринято
    grantedWithMods         (1), -- представлено с Mods
    rejection               (2), -- отвергнуто
    waiting                 (3), -- ожидание
    revocationWarning       (4), -- предупреждение об отзыве
    revocationNotification  (5), -- уведомление об отзыве
    keyUpdateWarning        (6)  -- предупреждение о обновлении ключа
}
```

Для обеспечения дополнительной информации в случаях отказов отвечающая сторона может использовать показанный ниже синтаксис.

```
PKIFailureInfo ::= BIT STRING {
    badAlg              (0),
    badMessageCheck     (1),
    badRequest          (2),
    badTime             (3),
    badCertId           (4),
    badDataFormat       (5),
    wrongAuthority      (6),
    incorrectData       (7),
    missingTimeStamp    (8),
    badPOP              (9),
    certRevoked         (10),
    certConfirmed       (11),
    wrongIntegrity      (12),
    badRecipientNonce   (13),
    timeNotAvailable    (14),
    unacceptedPolicy    (15),
    unacceptedExtension (16),
    addInfoNotAvailable (17),
    badSenderNonce      (18),
    badCertTemplate     (19),
    signerNotTrusted    (20),
    transactionIdInUse  (21),
    unsupportedVersion  (22),
    notAuthorized       (23),
    systemUnavail       (24),
    systemFailure       (25),
    duplicateCertReq    (26)
}
PKIStatusInfo ::= SEQUENCE {
    status        PKIStatus,
    statusString  PKIFreeText     OPTIONAL,
    failInfo      PKIFailureInfo  OPTIONAL
}
```

5.2.4. Идентификация сертификата

Для идентификации отдельного сертификата служит структура данных CertId.

Синтаксис CertId описан в [CRMF].

5.2.5. «Автономный» открытый ключ корневого CA

Каждый корневой агент CA должен быть способен опубликовать свой текущий открытый ключ с помощью того или иного «автономного» (out-of-band) механизма. Хотя рассмотрение таких механизмов выходит за пределы данного документа, здесь определены структуры данных, которые могут поддерживать такие механизмы.

В общем случае доступны два метода - CA непосредственно публикует свой самоподписанный сертификат или эта информация делается доступной через службу каталогов (Directory) или эквивалентный сервис, а CA публикует хэш-сумму этого значения для обеспечения возможности проверки целостности перед использованием данных.

```
OOBCert ::= Certificate
```

Для полей этого сертификата вводится ряд ограничений:

- сертификат **должен** быть самоподписанным (т. е., подпись должна быть проверяемой с использованием поля SubjectPublicKeyInfo);
- поля subject и issuer **должны** быть идентичны;

- если subject = NULL, **должны** присутствовать оба расширения subjectAltNames и issuerAltNames, а их значения должны совпадать;

- значения всех остальных расширений должны быть применимыми к самоподписанным сертификатам (например, идентификаторы ключей для subject и issuer должны совпадать).

```
OOBCertHash ::= SEQUENCE {
        hashAlg     [0] AlgorithmIdentifier     OPTIONAL,
        certId      [1] CertId                  OPTIONAL,
        hashVal         BIT STRING
}
```

Назначение поля hashVal состоит в том, чтобы любая сторона, получившая защищённым способом (с использованием автономных механизмов) хэш-значение, могла проверить самоподписанный сертификат для данного CA.

5.2.6. Опции архивирования

Запрашивающая сторона может указывать своё пожелание к PKI об архивировании секретного ключа с использованием структуры PKIArchiveOptions.

Синтаксис PKIArchiveOptions описан в [CRMF].

5.2.7. Публикация данных

Запрашивающая сторона может указывать своё пожелание к PKI о публикации сертификата с использованием структуры PKIPublicationInfo.

Синтаксис PKIPublicationInfo описан в [CRMF].

5.2.8. Структуры PoP

Если запрос сертификата сделан для пары ключей, используемой при подписи (т. е., запрашивается сертификат для верификации) владение секретным ключом подписи демонстрируется с помощью структуры POPOSigningKey.

Синтаксис POPOSigningKey описан в Приложении C и [CRMF], но следует отметить, что данная спецификация накладывает некоторые условия на семантику POPOSigningKeyInput.

```
POPOSigningKeyInput ::= SEQUENCE {
        authInfo            CHOICE {
                        sender              [0] GeneralName,
                        publicKeyMAC            PKMACValue
        },
        publicKey           SubjectPublicKeyInfo
}
```

С другой стороны, если запрос сертификата сделан для пары ключей шифрования (т. е., запрашивается сертификат для шифрования), владение секретным ключом для дешифрования может быть продемонстрировано тремя описанными ниже способами.

5.2.8.1. Включение секретного ключа

Включение секретного ключа (в зашифрованном виде) в сообщение CertRequest (в поле thisMessage структуры POPOPrivKey (см. Приложение C) или PKIArchiveOptions в зависимости от того, требуется ли также архивирование секретного ключа).

5.2.8.2. Косвенный метод

Агент CA возвращает зашифрованный сертификат взамен обычного (т. е., сертификат шифруется со случайным значением симметричного ключа, а этот ключ шифруется с использованием открытого ключа, для которого сделан запрос сертификата) - этот метод ранее в параграфе 4.3.2 назван косвенным. Конечный элемент подтверждает агенту CA своё знание секретного ключа дешифровки путём включения корректного значения CertHash для данного сертификата в сообщение certConf. Это служит демонстрацией POP, поскольку EE может корректно рассчитать значение CertHash только в том случае, когда имеется возможность восстановить сертификат, а для этого требуется расшифровать симметричный ключ, используя соответствующий секретный ключ. Очевидно, что для того, чтобы такой механизм работал, агенту CA **недопустимо** публиковать сертификат, пока не будет получено сообщение certConf (в случае использования certHash для демонстрации POP). Более подробная информация приведена в параграфе 5.3.18.

5.2.8.3. Протокол «вызов-отклик»

Конечный элемент использует протокол «вызов-отклик» (на основе описанных ниже сообщений POPODecKeyChall и POPODecKeyResp) между структурами данных CertReqMessages и CertRepMessage - этот метод в параграфе 4.3.2 назван прямым[1]. Полная схема взаимодействия показана на рисунке (отметим, что сообщение req' не всегда инкапсулирует req, как вложенное сообщение.

Этот протокол обычно требует существенно больше времени, нежели 3-этапный обмен описанный в предыдущем параграфе, но позволяет использовать локальный агент RA, а сертификат реально не создаётся, пока не будет завершена процедура POP. В некоторых средах может использоваться иной порядок обмена, показанный на рисунке (выбор может определяться требованиями политики).

Если сертификат запрашивается для пары ключей KAK[2], в процедуре POP может использоваться любой из трёх описанных выше способов для пары ключей шифрования с некоторыми изменениями: (1) текст в скобках для параграфа 5.2.8.2 меняется на «(т. е., сертификат шифруется симметричным ключом, полученным из секретного ключа KAK агента CA и открытого ключа, для которого делается запрос сертификата)»; (2) текст в первых скобках

[1]Прямой метод обычно используется в средах, где агент RA верифицирует POP и после этого запрашивает сертификат у CA от имени конечного элемента. В таком сценарии CA доверяет RA в плане корректности проверки POP перед запросом агентом RA сертификата для конечного элемента.

[2]Key agreement key - ключ согласования ключей.

```
EE              RA              CA
 ---- req ---->
 <--- chall ---
 ---- resp --->
                  ---- req' --->
                  <--- rep -----
 <--- rep -----
 ---- conf --->
                  ---- conf --->
                  <--- ack -----
 <--- ack -----
```

приведённого ниже описания поля challenge[1] меняется на «(с использованием PreferredSymmAlg (см. параграф 5.3.19.4 и Приложение E.5) и симметричного ключа, полученного из секретного ключа KAK агента CA и открытого ключа, для которого делается запрос сертификата)». В дополнение к этому для POP можно использовать структуру POPOSigningKey, описанную в [CRMF] (поле alg имеет значение DHBasedMAC, а поле signature — MAC), в качестве четвёртого варианта, если CA уже имеет сертификат D-H, известный EE.

Сообщения «вызов-отклик» процедуры POP для секретного ключа дешифровки описаны ниже (см. также стр. 404 документа [MvOV97]). Отметим, что этот обмен «вызов-отклик» связан с предшествующим сообщением для запроса сертификата (а также последующим откликом на такой запрос и подтверждением) через идентификатор transactionID в заголовке PKIHeader и защиту (MAC или подпись), применяемую к сообщению PKIMessage.

```
POPODecKeyChallContent ::= SEQUENCE OF Challenge
Challenge ::= SEQUENCE {
        owf                 AlgorithmIdentifier  OPTIONAL,
        witness             OCTET STRING,
        challenge           OCTET STRING
}
```

Отметим, что размер Rand должен соответствовать требованиям шифрования с открытым ключом запрашивающей стороны. С учётом того, что размер int обычно не превышает 64 битов, остаётся более 100 байтов для поля sender, при использовании модуля в 1024 бита. Если в той или иной среде имена слишком велики и не могут поместиться (например, слишком длинные имена DN), следует использовать вмещающуюся часть (она должна обеспечивать получателю возможность однозначной трактовки сокращения).

```
POPODecKeyRespContent ::= SEQUENCE OF INTEGER
```

5.2.8.4. Обзор опций PoP

В этом параграфе описано несколько опций, связанных с методами POP. Список методов (сокращение SK[2] означает ключ подписи, EK[3] - ключ шифрования, KAK - ключ согласования ключей) приведён ниже.

```
RAVerified;
SKPOP;
EKPOPThisMessage;
KAKPOPThisMessage;
KAKPOPThisMessageDHMAC;
EKPOPEncryptedCert;
KAKPOPEncryptedCert;
EKPOPChallengeResp;
KAKPOPChallengeResp.
```

С учётом этого массива опций возникает вопрос - как конечный элемент может узнать набор опций, поддерживаемых агентом CA/RA (т. е., понять, какие опции могут применяться при запросе сертификатов). Приведённые ниже рекомендации проясняют этот вопрос для разработчиков EE.

RAVerified. Выбор этой опции не определяется EE - агент RA использует эту опцию тогда и только тогда, когда он проверяет POP перед пересылкой запроса агенту CA, следовательно, EE не может выбирать этот метод.

SKPOP. Если EE имеет пару ключей подписи, этот вариант является единственным методом POP, заданным данной спецификацией для использования при запросе соответствующего сертификата.

EKPOPThisMessage и KAKPOPThisMessage. Конечный элемент сам принимает решение о предоставлении своего секретного ключа агенту CA/RA. Если EE считает нужным открыть свой ключ, этот вариант является единственным методом POP, заданным данной спецификацией для решения этой задачи (выбор одного из двух методов будет определяться типом ключевой пары).

KAKPOPThisMessageDHMAC. EE может использовать этот метод только при выполнении двух условий - (1) агент CA имеет сертификат DH, подходящий для этой цели, и (2) EE уже имеет копию этого сертификата. При выполнении обоих условий этот метод поддерживается явно и может быть использован конечным элементом.

EKPOPEncryptedCert, KAKPOPEncryptedCert, EKPOPChallengeResp, KAKPOPChallengeResp. EE помещает одну из этих опций (в зависимости от предпочтений и типа ключа) в поле subsequentMessage запросного сообщения. Тем самым, EE не выполняет процедуры POP, а просто указывает желаемый метод. Следовательно, если CA/RA возвращает сообщение об ошибке badPOP, конечный элемент повторяет запрос с использованием другого метода POP в сообщении subsequentMessage. Отметим, однако, что данная спецификация поощряет использование методов EncryptedCert и, более того, говорит, что «вызов-отклик» будет применяться обычно при вовлечении RA в верификацию POP. Таким образом, EE следует поддерживать возможность принятия интеллектуального решения о выборе одного из этих методов POP в сообщении-запросе.

[1] Текст исходного документа не позволяет однозначно определить, в каком месте заменяется текст. *Прим. перев.*
[2] Signing key.
[3] Encryption key.

5.3. Структуры данных, связанные с операциями

5.3.1. Запрос инициализации

Запрос Initialization содержит в качестве PKIBody структуру данных CertReqMessages, которая задаёт запрашиваемый(е) сертификат(ы). Обычно поля SubjectPublicKeyInfo, KeyId и Validity являются шаблонами, которые могут быть применены к каждому сертификату (см. профили в Приложении D). Этот тип сообщений предназначен для использования при первичной инициализации в PKI.

Синтаксис CertReqMessages описан в Приложении C и документе [CRMF].

5.3.2. Отклик при инициализации

Отклик Initialization содержит в качестве PKIBody структуру данных CertRepMessage, которая включает поле PKIStatusInfo для каждого из запрошенных сертификатов, а также может включать секретный ключ (обычно он шифруется сеансовым ключом, который, в свою очередь, шифруется ключом protocolEncrKey).

Синтаксис CertRepMessage описан в параграфе 5.3.4. Отметим, что при использовании для защиты сообщений PKI разделяемого секрета (см. параграф 5.1.3) любому сертификату, передаваемому в поле caPubs, инициатор может доверять непосредственно, как сертификату корневого CA.

5.3.3. Запрос сертификации

Запросное сообщение Certification содержит в качестве PKIBody структуру данных CertReqMessages, которая задаёт запрашиваемые сертификаты. Это сообщение предназначено для использования действующими элементами PKI, когда им нужны дополнительные сертификаты.

Синтаксис CertReqMessages описан в Приложении C и документе [CRMF].

В дополнение к этому PKIBody **может** представлять собой структуру данных CertificationRequest (эта структура полностью в формате ASN.1 в документе [PKCS10]). Данная структура может потребоваться при запросах сертификатов для ключей подписи при необходимости организации взаимодействия с унаследованными системами, но использовать её без крайней необходимости настоятельно не рекомендуется.

5.3.4. Отклик при сертификации

Отклик Certification содержит в качестве PKIBody структуру данных CertRepMessage, которая включает значение статуса для каждого запрошенного сертификата и может включать также открытый ключ CA, информацию об отказах, сертификат субъекта и зашифрованный секретный ключ.

```
CertRepMessage ::= SEQUENCE {
        caPubs                  [1] SEQUENCE SIZE (1..MAX) OF Certificate
                                OPTIONAL,
        response                SEQUENCE OF CertResponse
}
CertResponse ::= SEQUENCE {
        certReqId           INTEGER,
                status              PKIStatusInfo,
                certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
                rspInfo             OCTET STRING        OPTIONAL
                --аналогично строке id-regInfo-utf8Pairs, определённой для regInfo в CertReqMsg [CRMF]
}
CertifiedKeyPair ::= SEQUENCE {
        certOrEncCert       CertOrEncCert,
                privateKey      [0] EncryptedValue      OPTIONAL,
                -- см. комментарии по кодированию в [CRMF]
                publicationInfo [1] PKIPublicationInfo  OPTIONAL
}
CertOrEncCert ::= CHOICE {
        certificate     [0] Certificate,
                encryptedCert   [1] EncryptedValue
}
```

В каждой структуре CertResponse может присутствовать только одно из полей (в зависимости от статуса результата) - failInfo (в PKIStatusInfo) или certificate (в CertifiedKeyPair). Для некоторых значений статуса (например, waiting - ожидание) не будет присутствовать ни одного из этих полей.

С помощью EncryptedCert и соответствующего ключа расшифровки сертификат может быть получен. Целью является обеспечение агенту CA возможности возврата значения сертификата таким образом, чтобы актуальный сертификат был доступен только предусмотренному получателю. Преимущество такого подхода заключается в том, что CA может передавать сертификат в отклике даже при отсутствии уверенности в том, что запрашивающая сторона является именно тем конечным элементом, который может использовать подходящий секретный ключ (отметим, что подтверждения не будет, пока агент CA не получит сообщения certConf). Таким образом, агенту CA не требуется отзывать сертификат при возникновении какой-либо ошибки при проверке владения (но агент **может** отозвать сертификат, если этого требует политика).

5.3.5. Содержимое запроса на обновление ключа

Для запросов на обновление ключей используется синтаксис CertReqMessages. Обычно SubjectPublicKeyInfo, KeyId и Validity являются шаблонами полей, которые могут быть представлены для каждого обновляемого ключа. Этот тип сообщений предназначен для запроса на обновление действующих (не отозванных и не просроченных) сертификатов (поэтому иногда его называют обновлением сертификатов - Certificate Update). Обновление является заменой сертификата, содержащей новый новый или текущий открытый ключ субъекта (второй вариант может быть неприемлемым для некоторых сред).

Синтаксис CertReqMessages описан в Приложении C и документе [CRMF].

5.3.6. Содержимое отклика при обновлении ключа

Для откликов на запрос обновления ключей используется синтаксис CertRepMessage. Отклик идентичен отклику при инициализации.

Синтаксис CertRepMessage описан в параграфе 5.3.4.

5.3.7. Содержимое запроса на восстановление ключа

Для запросов на восстановление ключей используется синтаксис, аналогичный инициализационному запросу CertReqMessages. Обычно поля SubjectPublicKeyInfo и KeyId являются шаблонами, которые могут быть использованы для открытого ключа подписи, для которого запрашивается сертификат (см. также Приложение D).

Синтаксис CertReqMessages описан в Приложении C и документе [CRMF]. Отметим, что при наличии необходимости в истории ключа запрашивающая сторона должна включить в сообщение элемент управления Protocol Encryption Key.

5.3.8. Содержимое отклика при восстановлении ключа

Синтаксис для откликов при восстановлении ключей показан ниже. Для некоторых значений статуса (например, waiting - ожидание) все необязательные поля могут отсутствовать.

```
KeyRecRepContent ::= SEQUENCE {
        status              PKIStatusInfo,
                newSigCert  [0]       Certificate                                  OPTIONAL,
                caCerts     [1]       SEQUENCE SIZE (1..MAX) OF Certificate        OPTIONAL,
                keyPairHist [2]       SEQUENCE SIZE (1..MAX) OF CertifiedKeyPair   OPTIONAL
}
```

5.3.9. Содержимое запроса на отзыв

При запросе на отзыв сертификата (или нескольких сертификатов) используется приведённая ниже структура данных. Имя запрашивающей стороны присутствует в структуре PKIHeader.

```
    RevReqContent ::= SEQUENCE OF RevDetails

    RevDetails ::= SEQUENCE {
        certDetails         CertTemplate,
        crlEntryDetails     Extensions       OPTIONAL
    }
```

5.3.10. Содержимое отклика при отзыве

Отклик при отзыве является ответом на описанное выше сообщение. Если такой отклик создаётся, он направляется стороне, запросившей отзыв (отдельное сообщение, анонсирующее отзыв сертификата может быть передано субъекту, для которого запрошен отзыв сертификата).

```
RevRepContent ::= SEQUENCE {
        status         SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
        revCerts  [0]  SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL,
        crls      [1]  SEQUENCE SIZE (1..MAX) OF CertificateList    OPTIONAL
}
```

5.3.11. Содержимое запроса кросс-сертификата

Для запросов кросс-сертификации используется такой же синтаксис (CertReqMessages), как при обычном запросе сертификата, но ключевая пара **должна** генерироваться запрашивающим агентом CA, а секретный ключ **недопустимо** передавать отвечающему агенту CA. Такой запрос **может** использоваться также дочерними CA для получения своих сертификатов, подписанных родительским агентом CA.

Синтаксис CertReqMessages описан в Приложении C и документе [CRMF].

5.3.12. Содержимое отклика при кросс-сертификации

В откликах при кросс-сертификации используется такой же синтаксис (CertRepMessage), при обычных откликах на запрос сертификата, но зашифрованный секретный ключ в них передаваться не может.

Синтаксис CertRepMessage описан в параграфе 5.3.4.

5.3.13. Содержимое анонса обновления ключа CA

Когда агент CA обновляет свою пару ключей, для анонса **может** использоваться показанная ниже структура данных.

```
CAKeyUpdAnnContent ::= SEQUENCE {
        oldWithNew          Certificate,
        newWithOld          Certificate,
        newWithNew          Certificate
}
```

5.3.14. Анонсирование сертификата

Описанная здесь структура **может** применяться для анонсирования существования сертификатов.

Отметим, что это сообщение предназначено для использования в тех случаях (если они возникают), когда нет заранее созданного метода публикации сертификатов, и не предназначено для применения в тех случаях, когда имеется метод публикации сертификатов (например, X.500).

```
        CertAnnContent ::= Certificate
```

5.3.15. Анонсирование отзыва

Когда агент CA отзывает конкретный сертификат или близится срок отзыва, агент **может** анонсировать это (возможно, грядущее) событие.

```
RevAnnContent ::= SEQUENCE {
        status              PKIStatus,
        certId              CertId,
        willBeRevokedAt     GeneralizedTime,
        badSinceDate        GeneralizedTime,
        crlDetails          Extensions       OPTIONAL
}
```

CA **может** использовать этот анонс для предупреждения (или уведомления) субъекта, чей сертификат отзывается (или будет отозван). Обычно это происходит в тех случаях, когда запрос на отзыв исходит не от субъекта, чей сертификат будет отзываться.

Поле willBeRevokedAt содержит значение времени, когда в соответствующий список CRL была внесена новая запись.

5.3.16. Анонсирование CRL

Когда агент CA создаёт новый список CRL (или набор CRL) для анонсирования этого события **может** использоваться приведённая ниже структура.

```
        CRLAnnContent ::= SEQUENCE OF CertificateList
```

5.3.17. Содержимое подтверждения PKI

Эта структура данных используется в протокольном обмене, как финальное сообщение PKIMessage. Содержимое этого сообщения одинаково для всех случаев - реально никакого содержимого нет, поскольку вся требуемая информация содержится в заголовке PKIHeader.

```
        PKIConfirmContent ::= NULL
```

Использовать эти сообщения для подтверждения сертификатов **не рекомендуется**, **следует** вместо этого применять certConf. При получении PKIConfirm в отклике для сертификата получатель **может** трактовать его, как certConf для подтверждения восприятия всех сертификатов.

5.3.18. Содержимое подтверждения сертификата

Эта структура данных используется клиентом для передачи агенту CA/RA подтверждения о восприятии или отказе от сертификата.

```
CertConfirmContent ::= SEQUENCE OF CertStatus

CertStatus ::= SEQUENCE {
        certHash    OCTET STRING,
        certReqId   INTEGER,
        statusInfo  PKIStatusInfo     OPTIONAL
}
```

Для любого конкретного значения CertStatus опущенное поле statusInfo показывает **восприятие** указанного сертификата. Кроме того, **могут** явно указываться (как для восприятия, так и для отказа) в поле statusInfo дополнительные сведения о результате (например, для системы аудита агента CA/RA).

Внутри CertConfirmContent отсутствие структуру CertStatus, соответствующей сертификату, представленному в предшествующем отклике, говорит о том, что сертификат **отвергнут**. Поэтому **может** использоваться пустое значение CertConfirmContent (SEQUENCE нулевого размера) для индикации отказа от всех представленных сертификатов. В параграфе 5.2.8 (п. 2) приведено обсуждение поля certHash применительно к подтверждению владения.

5.3.19. Содержимое сообщений PKI общего назначения

```
InfoTypeAndValue ::= SEQUENCE {
        infoType               OBJECT IDENTIFIER,
        infoValue              ANY DEFINED BY infoType  OPTIONAL
}
-- где {id-it} = {id-pkix 4} = {1 3 6 1 5 5 7 4}
GenMsgContent ::= SEQUENCE OF InfoTypeAndValue
```

5.3.19.1. Сертификат шифрования протокола CA

Может использоваться конечным элементом для получения от агента CA сертификата с целью защиты информации в процессе работы протокола.

```
      GenMsg:    {id-it 1}, < absent >
      GenRep:    {id-it 1}, Certificate | < absent >
```

Конечные элементы **должны** гарантировать корректность сертификата, используемого для этой цели.

5.3.19.2. Типы ключевых пар для подписи

Может использоваться конечным элементом для получения списка алгоритмов цифровой подписи (например, RSA, DSA), для которых агент CA будет сертифицировать значения открытых ключей субъекта. Отметим, что для целей обмена сообщениями значения rsaEncryption и rsaWithSHA1, например, рассматриваются, как эквивалентные; вопрос можно сформулировать так: «Желает ли CA сертифицировать открытый ключ RSA?»

```
      GenMsg:    {id-it 2}, < absent >
      GenRep:    {id-it 2}, SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier
```

5.3.19.3. Типы ключевых пар для шифрования/согласования ключей

Может использоваться клиентом для получения списка алгоритмов шифрования/согласования ключей, для которых агент CA будет сертифицировать открытые ключи субъекта.

```
GenMsg:    {id-it 3}, < absent >
GenRep:    {id-it 3}, SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier
```

5.3.19.4. Предпочтительный симметричный алгоритм

Может использоваться клиентом для получения информации о предпочтительном для агента CA симметричном алгоритме шифрования, который будет нужен при обмене данными между EE и CA (например, когда EE хочет передать свой секретный ключ агенту CA с целью архивирования).

```
GenMsg:    {id-it 4}, < absent >
GenRep:    {id-it 4}, AlgorithmIdentifier
```

5.3.19.5. Обновление ключевой пары CA

Может использоваться агентом CA для анонсирования обновления ключа.

```
GenMsg:    {id-it 5}, CAKeyUpdAnnContent
```

5.3.19.6. CRL

Может использоваться клиентом для получения копии последнего списка отозванных сертификатов (CRL).

```
GenMsg:    {id-it 6}, < absent >
GenRep:    {id-it 6}, CertificateList
```

5.3.19.7. Неподдерживаемые идентификаторы объекта

Используется сервером для возврата списка идентификаторов объектов, которые не распознаны или не поддерживаются, но включены в представленный клиентом список.

```
GenRep:    {id-it 7}, SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER
```

5.3.19.8. Параметры ключевой пары

Может использоваться конечным элементом для запроса параметров домена, применяемых при генерации ключевых пар для некоторых алгоритмов с открытым ключом. Может служить, например, для запроса подходящих значений P, Q и G при генерации ключа DH/DSA или для запроса общеизвестных эллиптических кривых.

```
GenMsg:    {id-it 10}, OBJECT IDENTIFIER -- (идентификатор объекта в алгоритме)
GenRep:    {id-it 11}, AlgorithmIdentifier | < absent >
```

Отсутствие infoValue в GenRep показывает, что заданный в GenMsg алгоритм не поддерживается.

Конечные элементы **должны** гарантировать, что эти параметры приемлемы для них, а сообщение GenRep заверено (для предотвращения атак с подменой).

5.3.19.9. Пароль для отзыва

Может использоваться конечным элементом для передачи пароля агенту CA/RA с целью заверения последующего запроса на отзыв (в случаях, когда соответствующий секретный ключ уже невозможно использовать для заверения запроса). Дополнительная информация об использовании этого механизма приведена в Приложении B.

```
GenMsg:    {id-it 12}, EncryptedValue
GenRep:    {id-it 12}, < absent >
```

5.3.19.10. ImplicitConfirm

Определение и описание применения {id-it 13} дано в параграфе 5.1.1.1.

5.3.19.11. ConfirmWaitTime

Определение и описание применения {id-it 14} дано в параграфе 5.1.1.2.

5.3.19.12 Исходное сообщение PKIMessage

Определение и описание применения {id-it 15} дано в параграфе 5.1.3.

5.3.19.13. Поддерживаемые теги языка

Может использоваться для определения тега языка, подходящего для использования в последующих сообщениях. Отправитель передаёт список поддерживаемых языков (в порядке снижения уровня предпочтения), а получатель возвращает подходящий для использования тег. (Отметим, что каждая строка UTF8String **должна** включать тег языка.) Если ни один из предложенных языков не поддерживается получателем сообщения, он **должен** возвратить сообщение об ошибке.

```
GenMsg:    {id-it 16}, SEQUENCE SIZE (1..MAX) OF UTF8String
GenRep:    {id-it 16}, SEQUENCE SIZE (1) OF UTF8String
```

5.3.20. Содержимое отклика PKI общего назначения

```
GenRepContent ::= SEQUENCE OF InfoTypeAndValue
```

Примеры GenRep, которые **могут** поддерживаться, включают те, что перечислены в подпараграфах параграфа 5.3.19.

5.3.21. Содержимое сообщений об ошибках

Эта структура данных **может** использоваться EE, CA и RA для передачи информации об ошибке.

```
ErrorMsgContent ::= SEQUENCE {
        pKIStatusInfo          PKIStatusInfo,
        errorCode              INTEGER           OPTIONAL,
        errorDetails           PKIFreeText       OPTIONAL
}
```

Это сообщение **может** генерироваться в любой момент на протяжении транзакции PKI. Когда клиент передаёт такой запрос, сервер **должен** возвратить в ответ сообщение PKIConfirm или другое сообщение ErrorMsg, если какая-либо часть заголовка является некорректной. Обе стороны **должны** трактовать такое сообщение, как завершение транзакции (если таковая происходит).

Если для сообщения нужна защита, клиент **должен** использовать для этого тот же метод (например, подпись или MAC), что применялся для стартового сообщения данной транзакции. Агент CA всегда **должен** подписывать сообщение своим ключом подписи.

5.3.22. Запрос и отклик при опросе

Эта пара сообщений предназначена для обслуживания ситуаций, когда клиенту нужно опросить сервер для определения статуса незавершённых транзакций ir, cr или kur (т. е., при «ожидании» получения PKIStatus).

```
PollReqContent ::= SEQUENCE OF SEQUENCE {
       certReqId    INTEGER }

PollRepContent ::= SEQUENCE OF SEQUENCE {
       certReqId    INTEGER,
       checkAfter   INTEGER,  -- время в секундах
       reason       PKIFreeText OPTIONAL }
```

Ниже перечислены ситуации, когда используется опрос и способы применения. Предполагается, что во время транзакции может быть передано множество сообщений certConf. Одно такое сообщение будет передаваться в ответ на каждое сообщение ip, cp или kup, содержащее CertStatus для эмиттированного сертификата.

1. В ответ на сообщение ip, cp или kup конечный элемент будет передавать certConf для всех эмиттированных сертификатов и, вслед за этим, pollReq для всех ожидаемых сертификатов.
2. В ответ на pollReq агент CA/RA будет возвращать ip, cp или kup, если один или несколько ожидаемых сертификатов уже готовы; в противном случае будет возвращаться pollRep.
3. Если EE получает pollRep, он будет ждать в течение времени не менее заданного полем checkAfter перед отправкой другого pollReq.
4. Если сообщение ip, cp или kup получено в ответ на pollReq, оно трактуется так же, как при начальном отклике.

В приведённом ниже примере конечный элемент включает для сертификата в один запрос.

Этап	*Конечный элемент*				*PKI*
1	Форматирование ir				
2		->	ir	->	
3					Обработка ir
4					При необходимости участие оператора для обоих сертификатов
5		<-	ip	<-	
6	Обработка ip				
7	Форматирование pReq				
8		->	pReq	->	
9					Проверка статуса запросов на сертификаты
10					Сертификаты не готовы
11					Форматирование pRep
12		<-	pRep	<-	
13	Ожидание				
14	Форматирование pReq				
15		->	pReq	->	
16					Проверка статуса запросов на сертификаты
17					Один сертификат готов
18					Форматирование ip
19		<-	ip	<-	
20	Обработка ip				
21	Форматирование certConf				
22		->	certConf	->	
23					Обработка certConf
24					Форматирование ack

```
25                          <-     pkiConf  <-
26  Форматирование pReq     ->     pReq     ->
27                                             Проверка статуса запросов на сертификаты
28                                             Сертификат готов
29                                             Форматирование ip
30                          <-     ip       <-
31  Обработка ip
32  Форматирование certConf
33                          ->     certConf ->
34                                             Обработка certConf
35                                             Форматирование ack
36                          <-     pkiConf  <-
```

6. Обязательные функции управления PKI

Некоторые из функций управления PKI, обозначенные в параграфе 3.1, более подробно описываются здесь.

В этом разделе рассматриваются функции, которые являются обязательными в том смысле, что каждая реализация конечного элемента и агента CA/RA **должна** обеспечивать соответствующую функциональность. Этот раздел, по сути, является профилем функциональности управления PKI, который **должен** поддерживаться. Отметим, однако, что функции управления, описанные в этом разделе, не требуется реализовывать с использованием описанных в разделе 5 сообщений PKI, если в данной среде доступны другие варианты (см. Приложение D, где описаны профили сообщений PKIMessage, которые **должны** поддерживаться).

6.1. Инициализация корневого CA

[Определение корневого CA приведено в параграфе 3.1.1.2 этого документа.]

Вновь созданный корневой агент CA должен выполнить «само-сертификацию», которая обеспечивается структурой Certificate с профилем, определенным для сертификата newWithNew, эмиттируемого вслед за обновлением ключа корневого CA.

Для того, чтобы самосертификация CA была доступна конечным элементам, которые не получают этот сертификат по «автономным» каналам, агент CA должен также создавать оттиск этого сертификата. Конечные элементы, которые получают сертификат защищёнными «автономными» способами, смогут верифицировать самосертификацию CA и, следовательно, другие атрибуты.

Для передачи оттиска используется структура данных OOBCertHash.

6.2. Обновление ключа корневого CA

Ключи CA (как и все другие ключи) имеют конечное время жизни и должны периодически обновляться. Агент CA **может** эмиттировать сертификаты NewWithNew, NewWithOld и OldWithNew (см. параграф 4.4.1), чтобы помочь существующим конечным элементам, которые используют текущий самоподписанный сертификат CA (OldWithOld), в защищённом переходе к новому самоподписанному сертификату CA (NewWithNew), а также новым конечным элементам, которые будут владеть сертификатом NewWithNew, в защищённом получении сертификата OldWithOld для верификации существующих данных.

6.3. Инициализация второстепенного CA

[Определение термина «второстепенный агент CA» приведено выше в параграфе 3.1.1.2.]

С точки зрения протоколов управления PKI инициализация второстепенного агента CA не отличается от инициализации конечного элемента. Единственное различие заключается в том, что второстепенный агент CA должен также создавать начальный список отзыва.

6.4. Создание CRL

Перед выпуском любого сертификата вновь созданный агент CA (который эмиттирует CRL) должен создать «пустые» версии каждого списка, который он будет периодически создавать.

6.5. Запрос информации PKI

Когда элемент PKI (CA, RA или EE) хочет получить информацию о текущем статусе CA, он **может** передать агенту CA запрос на такую информацию.

CA **должен** ответить на запрос, предоставив (по крайней мере) всю информацию, указанную в запросе. Если часть информации не может быть предоставлена, запрашивающей стороне должно быть возвращено сообщение об ошибке.

Если для запроса и предоставления информации PKI используются сообщения PKIMessage, запрос **должен** представлять собой сообщение GenMsg, отклик **должен** быть сообщением GenRep, а информация об ошибке **должна** быть сообщением Error. Эти сообщения защищают с использованием MAC на основе разделяемого секрета (т. е., PasswordBasedMAC) или иных заверенных механизмов (если конечный элемент имеет действующий сертификат).

6.6. Кросс-сертификация

Запрашивающий CA - это агент CA, который является субъектом кросс-сертификата; отвечающим CA будет агент, эмиттирующий кросс-сертификат.

Запрашивающий агент CA должен находиться в рабочем состоянии на момент запроса кросс-сертификации.

6.6.1. Односторонняя схема «запрос-отклик»

Схема кросс-сертификации по своей природе является односторонней — при успешном завершении результатом этой операции является создание одного нового кросс-сертификата. Если требуется организовать «двухстороннюю» кросс-сертификацию, каждый из агентов CA, в свою очередь, должен инициировать запрос кросс-сертификата (или использовать иную схему).

Описываемая схема подходит для случаев, когда оба участвующих в процессе агента CA уже верифицировали взаимно подписи друг друга (или имеют некую общую доверенную точку) или имеется возможность автономной верификации запрашивающей сертификат стороны.

Описание.

Кросс-сертификация инициируется запрашивающим агентом CA. Администратор CA на отвечающей стороне идентифицирует CA, для которого запрошена кросс-сертификация, и оборудование отвечающего агента CA генерирует код авторизации. Администратор отвечающего CA передаёт этот код по автономному каналу администратору запрашивающего сертификат CA. Последний вводит код авторизации на запрашивающем CA для того, чтобы инициировать обмен по основному каналу.

Код авторизации используется в целях заверения о обеспечения целостности. Это осуществляется путём генерации симметричного ключа на основе кода авторизации и использования этого ключа для генерации кодов заверения (MAC) для всего обмена сообщениями. Заверить сообщения можно с использованием взамен MAC цифровой подписи, если агенты CA способны получить и проверить открытые ключи тем или иным способом (например, сравнением с хеш-значением, доставленным по автономному каналу).

Запрашивающий агент CA инициирует обмен путём генерации запроса на кросс-сертификат (ccr[1]) со «свежим» случайным значением (случайное значение запрашивающей стороны). Сообщение ccr передаётся отвечающему агенту CA. Поля этого сообщения защищают от изменения с помощью кода авторизации на основе MAC.

При получении сообщения отвечающий агент CA проверяет сообщение и MAC, сохраняет случайное значение и генерирует своё случайное число (случайное значение отвечающей стороны). После этого агент генерирует (и, при необходимости, архивирует) новый сертификат для запрашивающей стороны, содержащий открытый ключ запрашивающего CA и подписанный секретным ключом подписи отвечающего CA. Далее отвечающий агент CA передаёт отклик на запрос кросс-сертификата (ccp[2]). Поля этого сообщения защищают от изменения с помощью кода авторизации на основе MAC.

При получении сообщения ccp запрашивающий агент CA проверяет это сообщение (включая полученное в нем случайное число) и MAC. После этого запрашивающий CA отвечает сообщением certConf. Поля этого сообщения защищаются от изменения с помощью кода авторизации на основе MAC. Запрашивающий агент CA **может** записать полученный сертификат в репозиторий (Repository) для использования в последствии при создании пути для сертификата.

При получении сообщения certConf отвечающий агент CA проверяет это сообщение и MAC, а потом возвращает подтверждение, используя для этого сообщение PKIConfirm. Он **может** также опубликовать сертификат запрашивающей стороны, чтобы упростить в будущем создание пути.

Примечания.

1. Сообщение ccr должно содержать «полный» запрос сертификата, т. е., запрашивающим агентом должны быть заданы все поля за исключением порядкового номера (включая, например, расширение BasicConstraints).
2. В сообщение ccp **следует** включать сертификат верификации отвечающего CA; при наличии такого сертификата запрашивающий агент CA должен проверить его (например, с помощью «автономного» механизма).

(Можно представить себе также простую, неинтерактивную модель кросс-сертификации, в которой эмиттирующий сертификат агент CA получает открытый ключ CA-субъекта из некого репозитория, верифицирует ключ с использованием автономного механизма, а после этого создаёт и публикует кросс-сертификат без явного вовлечения в процесс CA-субъекта. Эта модель может быть достаточно легитимной для множества сред, но, поскольку она не требует никакого обмена протокольными сообщениями, её рассмотрение выходит за рамки данной спецификации.)

6.7. Инициализация конечного элемента

Как и агенты CA, конечные элементы требуется инициализировать. Инициализация конечного элемента требует, по крайней, мере двух шагов:

- получение информации PKI;
- автономная верификация открытого ключа корневого агента CA.

(к возможным этапам инициализации относятся также получение информации об условиях доверия и/или автономная верификация открытых ключей других агентов CA).

6.7.1. Получение информации PKI

Требуемая информация включает:

- открытый ключ текущего корневого CA;
- (если сертифицирующий CA не является корневым) сертификационный путь от корневого CA к сертифицирующему агенту CA вместе с соответствующими списками отзыва;
- алгоритмы, поддерживаемые сертифицирующим CA, и их параметры для каждого варианта применения.

[1] Cross-certification request.
[2] Cross-certification response.

Может потребоваться дополнительная информация (например, поддерживаемые расширения или сведения о политике CA) для успешного выполнения сертификационного запроса. Однако для простоты мы не требуем обязательного получения такой информации конечными элементами с использованием сообщений PKI. Конечный результат состоит в том, что некоторые сертификационные запросы могут завершаться неудачно (например, конечный элемент хочет генерировать свой собственный ключ шифрования, но CA не позволяет делать это).

Требуемая информация может быть получена в соответствии с описанием в параграфе 6.5.

6.7.2. «Автономная» верификация ключа корневого CA

Конечный элемент может получить открытый ключ своего корневого CA защищённым путём. Одним из возможных способов служит предоставление конечному элементу самосертифицированного оттиска CA тем или иным «автономным» путём. Тогда конечный элемент сможет безопасно воспользоваться самосертификацией CA.

Дополнительная информация приведена в параграфе 6.1.

6.8. Запрос сертификата

Инициализированный конечный элемент **может** запросить дополнительный сертификат в любой момент (и с любой целью). Такой запрос осуществляется с использованием сообщения cr[1]. Если конечный элемент уже владеет ключевой парой для подписи (с соответствующим сертификатом верификации), сообщение cr обычно защищается с использованием цифровой подписи. Агент CA возвращает новый сертификат (при успешном выполнении запроса) в сообщении CertRepMessage.

6.9. Обновление ключа

Когда срок действия ключевой пары близок к завершению, соответствующий конечный элемент **может** запросить обновление ключей — т. е., он **может** запросить у агента CA выпуск нового сертификата для новой ключевой пары (или, в некоторых ситуациях, нового сертификата для прежней пары ключей). Запрос выполняется с помощью сообщения kur[2] (в некоторых средах это называют операцией обновления сертификата - Certificate Update). Если конечный элемент уже владеет ключевой парой для подписи (с соответствующим сертификатом верификации), сообщение kur обычно защищается с использованием цифровой подписи. Агент CA возвращает новый сертификат (при успешном выполнении запроса) в сообщении kup[3], синтаксически идентичном сообщению CertRepMessage.

7. Согласование версий

В этом разделе определено согласование версий между клиентом и серверов в целях поддержки ранних протоколов.

Если клиент знает версию(и) протокола, поддерживаемую сервером (например, из предшествующего обмена сообщениями PKIMessage или по автономному каналу), этот клиент **должен** передавать сообщения PKIMessage с максимальным номером версии, поддерживаемой им и сервером. Если клиент не знает поддерживаемой сервером версии(й), он **должен** передавать сообщения PKIMessage с использованием старшей версии, поддерживаемой им.

Если сервер получает сообщение поддерживаемой им версии, версия отклика **должна** совпадать с версией полученного сообщения. Если сервер получает сообщение, номер версии которого выше или ниже номера версии, поддерживаемого сервером, он **должен** возвратить сообщение ErrorMsg с установленным битом unsupportedVersion (в поле failureInfo структуры pKIStatusInfo). Если версия полученного сообщения выше старшей версии, поддерживаемой сервером, указываемая в сообщении об ошибке версия **должна** совпадать с максимальной версией, поддерживаемой сервером; если версия полученного сообщения ниже минимальной версии, поддерживаемой сервером, в сообщении об ошибке должна указываться минимальная версия из числа поддерживаемых сервером.

Если клиент получает сообщение ErrorMsgContent с установленным битом unsupportedVersion и поддерживаемым клиентом номером версии, он **может** повторить запрос с использованием указанной в сообщении об ошибке версии.

7.1. Поддержка реализаций RFC 2510

RFC 2510 не задаёт поведения реализаций при получении непонятных номеров версий, поскольку на момент выхода этого документа существовала лишь одна версия протокола. После введения настоящей спецификации рекомендуется описанное ниже поведение.

7.1.1. Взаимодействие клиентов с серверами RFC 2510

Если после передачи сообщения cmp2000 клиент получает ErrorMsgContent с версией cmp1999, он **должен** прервать текущую транзакцию. После этого клиент **может** повторить попытку с использованием сообщений версии cmp1999.

Если клиент получает не связанное с ошибкой сообщение PKIMessage версии cmp1999, он **может** принять решение о продолжении транзакции (если она ещё не закончена) с использованием семантики RFC 2510. Если клиент не хочет продолжать, а транзакция ещё не завершена, клиент **должен** прервать транзакцию и передать сообщение ErrorMsgContent с версией cmp1999.

7.1.2. Сервер, получающий сообщения cmp1999

Если сервер получает сообщение версии cmp1999, он **может** вернуться к поведению в стиле RFC 2510 и отвечать сообщениями версии cmp1999. Если сервер не пожелает снижать используемую версию, он **должен** возвратить сообщение ErrorMsgContent, как описано выше в параграфе 7.

[1] Certification request - запрос сертификации.
[2] Key update request - запрос на обновление ключей.
[3] Key update response - отклик на запрос обновления ключей.

8. Вопросы безопасности

8.1. Подтверждение владения ключом дешифровки

Некоторые криптографические вопросы заслуживают явного обсуждения. В описанных выше протоколах когда конечный элемент должен подтвердить владение ключом расшифровки, его фактически просят расшифровать что-либо (его собственный сертификат). Такая схема (и многие другие) может быть уязвима для атак, если рассматриваемого владельца ключа расшифровки можно обманным путём заставить расшифровывать некую произвольную информацию и возвращать отправителю вызова (атакующему) расшифрованный текст. Хотя в данной спецификации предусмотрено множество других средств для затруднения реализации таких атак, в некоторых будущих типах сервиса (например, нотариальных или точного времени) можно предположить наличие потенциальных уязвимостей для таких атак. По этой причине мы подтверждаем общее требование к реализациям соблюдать осторожность в части расшифровки произвольных данных и возврата расшифрованной информации, поскольку это может на практике приводить к возникновению серьёзных уязвимостей.

8.2. Подтверждение владения путём раскрытия секретного ключа

Отметим также, что раскрытие секретного ключа агенту CA/RA, как способ подтверждения владения, может приводить к возникновению угроз (в зависимости от того, насколько можно доверять CA/RA в плане корректного обслуживания такой информации). Разработчикам рекомендуется:

соблюдать осторожность при выборе и использовании этого метода POP;

по возможности получать от пользователей приложения явное подтверждение того, что они доверяют CA/RA иметь копию своего секретного ключа, прежде, чем раскрывать этот ключ.

8.3. Атаки на обмен ключами по методу Diffie-Hellman

В процессе обмена ключами по методу D-H возможна незначительная группа атак, описанных ниже. Враждебный конечный элемент преднамеренно выбирает параметры D-H, позволяющие ему получить секретный ключ D-H (или достаточно большую его часть) для агента CA при операциях архивирования и восстановления ключей. На основе этой информации EE сможет восстановить секретный ключ расшифровки другого конечного элемента (ничего не подозревающего об этом) EE2, в процессе легитимной операции по архивированию ключа EE2 на данном CA. Для предотвращения такой возможности есть два варианта. (1) CA может генерировать свежую пару ключей D-H для использования в качестве протокольной пары ключей шифрования каждому EE, с которым он взаимодействует. (2) CA может использовать протокол валидации ключей (не описан в этом документе) для каждого запрашивающего конечного элемента, чтобы гарантировать защищенность протокольной пары ключей шифрования EE от таких атак. Вариант(1) явно проще (не требует дополнительного протокольного обмена от какой-либо из сторон) и, следовательно, **рекомендован** к применению.

9. Взаимодействие с IANA

Типы сообщений PKI общего назначения различаются по идентификаторам объекта (OID). Значения OID для PKI General Message (сообщения общего назначения), определённые в этом документе, были выделены из сегмента, переданного IANA рабочей группе PKIX.

Упоминаемые в документе криптографические алгоритмы различаются по идентификаторам объектов (OID). Значения OID для криптографических алгоритмов были выделены из нескольких сегментов, принадлежащих разным организациям, включая RSA Security, Entrust Technologies, IANA и IETF.

Предполагается, что при вводе дополнительных алгоритмов шифрования значения OID для них будут выделяться из пространства предлагающей алгоритм организации.

От IANA не требуется выполнения каких-либо действий или внесения изменений в связи с данной спецификацией.

Нормативные документы

[X509] International Organization for Standardization and International Telecommunications Union, "Information technology - Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks", ISO Standard 9594-8:2001, ITU-T Recommendation X.509[1], March 2000.

[MvOV97] Menezes, A., van Oorschot, P. and S. Vanstone, "Handbook of Applied Cryptography", CRC Press ISBN 0-8493-8523-7, 1996.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2202] Cheng, P. and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202, September 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC2482] Whistler, K. and G. Adams, "Language Tagging in Unicode Plain Text", RFC 2482, January 1999.

[CRMF] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, September 2005.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[1]Этот стандарт на русском языке доступен по ссылке. *Прим. перев.*

Дополнительная литература

[CMPtrans] Kapoor, A., Tschalar, R. and T. Kause, "Internet X.509 Public Key Infrastructure -- Transport Protocols for CMP", Work in Progress. 2004.

[PKCS7] RSA Laboratories, "The Public-Key Cryptography Standards - Cryptographic Message Syntax Standard. Version 1.5", PKCS 7, November 1993.

[PKCS10] Nystrom, M., and B. Kaliski, "The Public-Key Cryptography Standards - Certification Request Syntax Standard, Version 1.7", RFC 2986, May 2000.

[PKCS11] RSA Laboratories, "The Public-Key Cryptography Standards - Cryptographic Token Interface Standard. Version 2.10", PKCS 11, December 1999.

[RFC1847] Galvin, J., Murphy, S., Crocker, S., and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[RFC2559] Boeyen, S., Howes, T. and P. Richard, "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2", RFC 2559, April 1999.

[RFC2585] Housley, R. and P. Hoffman, "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP", RFC 2585, May 1999.

[FIPS-180] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, May 1994.

[FIPS-186] National Institute of Standards and Technology, "Digital Signature Standard", FIPS PUB 186, May 1994.

[ANSI-X9.42] American National Standards Institute, "Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography", ANSI X9.42, February 2000.

Приложение A. Основания для присутствия RA

Причины присутствия агентов RA можно разделить на технические и организационные. Технические причины перечислены ниже.

- используются аппаратные маркеры, но не все конечные элементы имеют оборудование для их инициализации - оборудование RA может обеспечивать требуемую функциональность (это может быть также связано с политикой);
- некоторые конечные элементы не могут публиковать сертификаты - агент RA может делать это за них;
- RA имеет возможность эмиттировать подписанные запросы на отзыв от имени связанных с ним конечных элементов, тогда как конечные элементы могут не иметь такой возможности (если ключевая пара полностью потеряна).

Ниже перечислены некоторые причины организационного характера, обуславливающие присутствие RA.

- Концентрация функциональности в оборудовании RA может оказаться более экономичной по сравнению с реализацией тех же функций на всех конечных элементах (особенно при использовании специального оборудования для работы с аппаратными маркерами);
- организация RA может снизить число требуемых для работы агентов CA, что в некоторых случаях весьма желательно;
- RA может обеспечивать больше возможностей сопоставления людей с их «электронными» именами, особенно для случаев физической удалённости CA от конечного элемента;
- для многих приложений уже имеются те или иные административные структуры, поэтому кандидата на роль RA найти достаточно просто (для CA это может оказаться сложнее).

Приложение B. Использование пароля для отзыва

Запрос на отзыв должен содержать подходящие механизмы защиты, включая корректную идентификацию, для снижения вероятности успеха атак на службу (DoS). Цифровая подпись в запросе (она **обязательна** в соответствии с данной спецификацией, если поддерживаются запросы на отзыв) может обеспечить требуемую идентификацию, но при некоторых обстоятельствах могут быть желательны другие механизмы (например, когда секретный ключ больше не доступен, а конечный элемент желает запросить отзыв до ресертификации другой пары ключей). Для таких случаев данная спецификация также требует **обязательной** поддержки PasswordBasedMAC (в зависимости от локальной политики для данной среды), если поддерживаются запросы на отзыв и может быть организован разделяемый секрет для запрашивающей и отвечающей стороны до возникновения потребности в отзыве.

В некоторых средах используется механизм «отзыва по паролю» (revocation passphrase), в котором значение с достаточной энтропией (т. е., достаточно длинный пароль) совместно используется конечным элементом и CA/RA (и неизвестно другим) до запроса на отзыв - это значение может служить для заверения запроса на отзыв.

Данная спецификация включает несколько **опциональных** методов организации разделяемого секрета (т. е., пароля на отзыв). Их использование в сообщениях CMP описано ниже.

- OID и значение, заданное в параграфе 5.3.19.9, **могут** в любое время быть переданы в сообщении GenMsg или в поле generalInfo заголовка PKIHeader любого сообщения PKIMessage (в частности, значение EncryptedValue может быть передано в сообщении certConf, которое подтверждает приемлемость сертификатов, запрошенных в сообщении с запросом инициализации или сертификата). Они передают пароль на отзыв, выбранный элементом (т. е., зашифрованные байты поля encValue) соответствующему агенту CA/RA; более того, передача выполняется с использованием соответствующих параметров защиты конфиденциальности (поскольку пароль шифруется с помощью ключа protocolEncryptionKey агента CA/RA).

- Если агент CA/RA получает пароль для отзыва (OID и значение, заданное в параграфе 5.3.19.9) в сообщении GenMsg, он **должен** создать и передать сообщение GenRep, которое включает OID (без значения), как указано в параграфе 5.3.19.9. Если агент CA/RA получает пароль на отзыв в поле generalInfo заголовка PKIHeader в любом сообщении PKIMessage, он **должен** включить OID (без значения) в поле generalInfo заголовка PKIHeader соответствующего отклика PKIMessage. Если агент CA/RA не может возвратить требуемый отклик по той или иной причине, он **должен** передать сообщение об ошибке со статусом «отказ» (rejection), в которое может включить поле failInfo с описанием причины.

- Поле valueHint в EncryptedValue **может** содержать идентификатор ключа (выбирается элементом вместе с паролем), который будет помогать при последующем поиске корректного пароля (например, при создании запроса на отзыв элементом или при получении такого запроса агентом CA/RA).

- Сообщение с запросом на отзыв защищается с помощью PasswordBasedMAC, где пароль на отзыв служит ключом. В тех случаях, когда это приемлемо, поле senderKID в заголовке PKIHeader **может** содержать значение, переданное ранее в valueHint.

При использовании описанного выше метода пароль для отзыва может создаваться или изменяться без передачи избыточных сообщений или использования автономных методов. Например, само сообщение с запросом отзыва (защищённое и идентифицируемое с помощью кода MAC, который использует пароль в качестве ключа) может содержать в заголовке PKIHeader новый пароль для отзыва, который будет использоваться в последующих запросах на отзыв для любого из остальных сертификатов элемента. В некоторых средах это может оказаться более предпочтительным для механизмов, которые показывают пароль в запросе на отзыв, поскольку в таких случаях могут возникать атаки на службы, в которых раскрытый пароль может быть использован неуполномоченным третьим лицом для заверения запросов на отзыв других сертификатов элемента. Однако, поскольку пароль не раскрывается в запросах, здесь не предъявляется требования обновления пароля при каждом выполнении запроса (т. е., один пароль **может** использоваться элементом для заверения запросов на отзыв разных сертификатов в разное время).

Кроме того, описанный выше метод может обеспечивать строгую криптографическую защиту для всего сообщения с запросом на отзыв даже при отсутствии цифровой подписи. Методы, которые заверяют запросы просто путём раскрытия пароля на отзыв, не обеспечивают криптографической защиты для полей запроса (поэтому запрос на отзыв одного сертификата может быть изменён неуполномоченной стороной и использован для запроса на отзыв другого сертификата того же элемента).

Приложение C. Разъяснение поведения для запросов

При наличии расхождений с [CRMF], которые будут вызывать проблемы интерпретации или взаимодействия, [CRMF] следует трактовать, как нормативный документ.

Приведённые ниже определения заимствованы из [CRMF]. Эти определения включены для того, чтобы разъяснить поведенческие характеристики запросов; в остальных случаях синтаксис и семантика идентичны [CRMF].

```
CertRequest ::= SEQUENCE {
    certReqId     INTEGER,
    certTemplate  CertTemplate,
    controls      Controls OPTIONAL }

-- Если certTemplate представляет собой пустую последовательность SEQUENCE (т. е., все
-- поля опущены), поле controls МОЖЕТ содержать значение id-regCtrl-altCertTemplate,
-- задающее шаблон для сертификата, отличного от сертификата открытого ключа X.509v3
-- И наоборот, если значение certTemplate не пусто (т. е., заполнено хотя бы одно поле),
-- в поле НЕДОПУСТИМО включать значение id-regCtrl-altCertTemplate. Новый элемент
-- управления определяется следующим образом:

id-regCtrl-altCertTemplate OBJECT IDENTIFIER ::= {id-regCtrl 7}
AltCertTemplate ::= AttributeTypeAndValue

POPOSigningKey ::= SEQUENCE {
    poposkInput           [0] POPOSigningKeyInput OPTIONAL,
    algorithmIdentifier   AlgorithmIdentifier,
    signature             BIT STRING }

-- **********
-- * Для данной спецификации комментарий ASN.1, данный в [CRMF], относится не только к
-- * certTemplate, но и к altCertTemplate.
-- **********
-- * Подпись (с использованием algorithmIdentifier) является DER-представлением
-- * poposkInput (т. е., «значением» OCTETsof для POPOSigningKeyInput DER).
-- * Примечание: Если CertReqMsgcertReq certTemplate (или altCertTemplate) содержит
-- * значения subject и publicKey, поле poposkInput ДОЛЖНО быть опущено и подпись ДОЛЖНА
-- * рассчитываться для DER-представления значения CertReqMsg certReq (или
-- * AltCertTemplate). Если certTemplate/altCertTemplate не содержит значений субъекта
-- * и открытого ключа (т. е., содержится одно из этих значений или отсутствуют оба),
-- * поле poposkInput ДОЛЖНО присутствовать и ДОЛЖНО быть подписано.
-- **********

POPOPrivKey ::= CHOICE {
    thisMessage       [0] BIT STRING,

-- **********
-- * Тип thisMessage в [CRMF] указан, как строка битов (BIT STRING); следует использовать
-- * тип EncryptedValue (в соответствии с параграфом 5.2.2. Зашифрованные значения данной
-- * спецификации). Следовательно, данный документ разъясняет поведение, указывая, что
-- * содержимое поля thisMessage ДОЛЖНО кодироваться, как  EncryptedValue и после этого
-- * «преобразовываться» в BIT STRING. Такой подход обеспечивает требуемую транспортировку
```

```
-- * и защиту секретного ключа и совместимость на уровне битового представления с
-- * требованиями [CRMF].
-- **********

    subsequentMessage [1] SubsequentMessage,
    dhMAC             [2] BIT STRING }
```

Приложение D. Профили управляющих сообщений PKI (обязательны)

В этом приложении дано подробное описание профилей для тех сообщений PKIMessage, которые **должны** поддерживаться соответствующими спецификации реализациями (см. раздел 6).

Представлены профили для сообщений PKIMessage, используемых в операциях управления PKI:

- начальная регистрация/сертификация;
- схема с базовой идентификацией;
- запрос сертификата;
- обновление ключа.

D.1. Общие правила интерпретации профилей

1. Когда поле типа OPTIONAL или DEFAULT не упомянуто в индивидуальном профиле, его **не следует** включать в соответствующее сообщение (иначе получатель может отбросить такое сообщение, как синтаксически некорректное). Обязательные не указываются, если они имеют обычное значение (например, в данной версии спецификации pvno всегда принимает значение 2).
2. Когда структуры встречаются более, чем в одном сообщении, они по мере возможности указываются в профиле раздельно.
3. Идентификаторы алгоритмов из структуры PKIMessage указываются в профиле раздельно.
4. «Специальное» отличительное имя X.500 DN называется NULL-DN - это обозначает DN с последовательностью SEQUENCE OF RelativeDistinguishedNames нулевого размера (её DER-представление - 3000 в шестнадцатеричном формате).
5. В тех случаях, когда для поля требуется GeneralName, а подходящее значение недоступно (например, конечный элемент создаёт запрос до того, как станет известно его имя), для GeneralName используется значение X.500 NULL-DN (т. е., поле Name в CHOICE содержит NULL-DN). Этот специальный случай называют NULL-GeneralName.
6. Когда в профиле не указано значение GeneralName, в соответствующем поле сообщения PKIMessage указывается NULL-GeneralName. В частности, это относится к полю sender заголовка PKIHeader некоторых сообщений.
7. Там где может возникать неоднозначность в результате именования полей, в именах профилей используется нотация с точками (например, certTemplate.subject означает поле subject в поле с именем certTemplate).
8. Там, где SEQUENCE OF type является частью сообщения, используется нотация на основе массива с нулевой базой для описания полей в последовательности SEQUENCE OF (например, crm[0].certReq.certTemplate.subject указывает на субполе первого CertReqMsg, содержащегося в запросе).
9. Весь обмен сообщениями PKI, описанными в параграфах D.4 - D.6, требует передачи сообщений certConf элементом-инициатором и сообщений PKIConfirm с отвечающей стороны. Сообщения PKIConfirm не включены в некоторые профили, поскольку тело этих сообщений имеет значение NULL, а содержимое заголовка не включает контекста. Для protectionAlg могут использоваться любые идентифицированные способы (например, MAC на базе пароля, если известен разделяемый секрет, или цифровая подпись).

D.2. Алгоритм использования профиля

В приведённой ниже таблице даны определения алгоритмов, используемых в протоколах управления PKI. Таблица включает 4 колонки.

Имя: идентификатор, используемый для профилей сообщений;

Применение: описание цели и места использования алгоритма;

Обязательные: значения AlgorithmIdentifier, которые **должны** поддерживаться реализациями;

Прочие: дополнения к обязательным алгоритмам AlgorithmIdentifier.

Имя	*Применение*	*Обязательные*	*Прочие*
MSG_SIG_ALG	Защита сообщений PKI с использованием подписи	DSA/SHA-1	RSA/MD5, ECDSA, ...
MSG_MAC_ALG	Защита сообщений PKI с использованием MAC	PasswordBasedMac	HMAC, X9.9...
SYM_PENC_ALG	Симметричное шифрование секретного ключа конечного элемента в тех случаях, когда ключ симметричного шифрования распространяется автономно	3-DES (3-key-EDE, режим CBC)	AES, RC5, CAST-128...
PROT_ENC_ALG	Для шифрования (симметричных ключей, используемых для шифрования) секретных ключей, передаваемых в сообщениях PKIMessage, применяется асимметричный алгоритм	D-H	RSA, ECDH, ...
PROT_SYM_ALG	Для шифрования битов секретного ключа (ключи этого типа шифруются с помощью PROT_ENC_ALG) используется симметричный алгоритм	3-DES (3-key-EDE, режим CBC)	AES, RC5, CAST-128...

Идентификаторы и спецификации обязательных алгоритмов:

DSA/SHA-1:

AlgId: {1 2 840 10040 4 3};

Стандарт цифровой подписи [FIPS-186]

Размер открытого модуля: 1024 бита.

PasswordBasedMac:

AlgId: {1 2 840 113533 7 66 13} с SHA-1 {1 3 14 3 2 26} в качестве параметра owf и HMAC-SHA1 {1 3 6 1 5 5 8 1 2} в качестве параметра mac;

(данная спецификация) вместе с

Стандарт защищённого хэширования [FIPS-180] и [RFC2104]

Размер ключа HMAC: 160 битов (т. е., "K" = "H" в параграфе 5.1.3.1. Разделяемый секрет)

3-DES:

AlgId: {1 2 840 113549 3 7}; (используется в BSAFE алгоритма RSA и в S/MIME).

D-H:

AlgId: {1 2 840 10046 2 1};

[ANSI-X9.42]

Размер открытого модуля: 1024 бита.

```
DomainParameters ::= SEQUENCE {
   p       INTEGER, -- нечётный первичный, p=jq +1
   g       INTEGER, -- генератор, g^q = 1 mod p
   q       INTEGER, -- первичный множитель p-1
   j       INTEGER OPTIONAL, -- сомножитель, j>=2
   validationParms  ValidationParms OPTIONAL
}
ValidationParms ::= SEQUENCE {
   seed          BIT STRING, -- «затравка» для генерации первичного
   pGenCounter   INTEGER     -- верификация параметра
}
```

D.3. Подтверждение владения профилем

Ниже показаны поля POP для использования (в поле signature поля pop структуры ProofOfPossession) в случаях подтверждения владения секретным ключом подписи, соответствующим открытому ключу верификации, для которого был запрошен сертификат.

Поле	*Значение*	*Комментарий*
algorithmIdentifier	**MSG_SIG_ALG**	Для этого подтверждения разрешена только защита цифровой подписью.
signature	**присутствует**	Биты, рассчитанные с использованием MSG_SIG_ALG.

Для подтверждения владения секретным ключом дешифровки, соответствующим открытому ключу шифрования, для которого был запрошен сертификат, этот профиль не используется - взамен его применяется поле CertHash в сообщении certConf.

Не каждый агент CA/RA будет выполнять POP (для ключа подписи, ключа дешифровки или ключа согласования ключей) с использованием протокола запроса сертификатов по основному каналу PKIX-CMP (способ выполнения POP может, в конечном счёте, определяться политикой для любого заданного CA в его публикуемом Policy OID1 и Certification Practice Statement2). Однако данная спецификация **требует**, чтобы агенты CA/RA выполняли POP (тем или иным способом), как часть процесса сертификации. Все конечные элементы **должны** быть готовы обеспечить POP (т. е., эти компоненты протокола PKIX-CMP **должны** поддерживаться).

D.4. Начальная регистрация/сертификация (базовая схема)

(Инициализированный) конечный элемент запрашивает (первый) сертификат у агента CA. Когда CA отвечает сообщением, содержащим сертификат, конечный элемент отвечает подтверждением приёма сертификата. Агент CA возвращает сообщение PKIConfirm конечному элементу и завершает транзакцию. Все сообщения заверяются.

Такая схема позволяет конечному элементу запросить сертификацию для сгенерированного локально открытого ключа (обычно, ключ подписи). Конечный элемент **может** запросить централизованную генерацию и сертификацию другой пары ключей (обычно, пары ключей шифрования).

Сертификация может быть запрошена только для одного локально генерируемого открытого ключа (при большем количестве ключей используются раздельные сообщения PKIMessages).

Конечный элемент **должен** поддерживать подтверждение владения секретным ключом, связанным с локально сгенерированным открытым ключом.

Предпосылки:

1. конечный элемент может заверять подпись CA на основе автономных механизмов;
2. конечный элемент и CA совместно используют симметричный ключ MACing.

Поток сообщений:

№ этапа	*Конечный элемент*	*PKI*

```
1    Форматирование ir
2                                -> ir       ->
3                                                Обработка ir
4                                                Форматирование ip
5                                <- ip       <-
6    Обработка ip
7    Форматирование certConf
8                                -> certConf ->
9                                                Обработка certConf
10                                               Форматирование PKIConf
11                               <- PKIConf  <-
12   Обработка PKIConf
```

Для этого профиля вводится требование о том, что конечный элемент **должен** включать все (т. е., один или два) CertReqMsg в одно сообщение PKIMessage, а PKI (CA) **должен** генерировать одно сообщение PKIMessage, содержащее полный отклик (т. е., включающее **опциональную** вторую пару ключей, если та была запрошена и централизованная генерация ключей поддерживается). Для простоты спецификация также требует, что это сообщение **должно** быть окончательным (т. е., не использовалось значение статуса waiting ожидание).

Конечный элемент осуществляет также взаимодействие с агентом CA/RA по отдельному каналу (автономно). Эта транзакция создаёт разделяемый секрет, ссылочный номер referenceNumber и, **опционально**, отличительное имя, используемые для имен отправителя и субъекта в шаблоне сертификата. **Рекомендуется** использовать разделяемый секрет размером не менее 12 символов.

Запрос инициализации - ir

```
Поле                Значение
recipient           имя CA
  -- имя агента CA, у которого будет запрашиваться эмиссия сертификата
protectionAlg       MSG_MAC_ALG
  -- для этого запроса разрешена только защита MAC на основе ключа начальной идентификации
senderKID           referenceNum
  -- ссылочный номер, который ранее агент CA выдал для конечного элемента (вместе с ключом
  -- MACing)
transactionID       присутствует
  -- зависящее от реализации значение, не имеющее смысла для конечного элемента.
  -- [если идентификатор уже используется на CA, агент CA ДОЛЖЕН генерировать сообщение об
  -- отказе]
senderNonce         присутствует
  -- 128 (псевдо)случайных битов
freeText            любое корректное значение
body                ir (CertReqMessages)
                    допускается только 1 или 2 сообщения CertReqMsg
  -- если требуется большее число сертификатов, запросы ДОЛЖНЫ помещаться в отдельные
  -- сообщения PKIMessages

CertReqMsg          присутствует одно или два
  -- более подробное описание приведено ниже
  -- crm[0] обозначает первое (которое ДОЛЖНО присутствовать), а crm[1] — второе (которое
  -- является ОПЦИОНАЛЬНЫМ и служит для запроса централизованно генерируемого ключа)

crm[0].certReq.     Фиксированное значение 0
   certReqId
  -- индекс шаблона в сообщении
crm[0].certReq      присутствует
   certTemplate
  -- ДОЛЖНО включать значение открытого ключа субъекта, других ограничений нет
crm[0].pop...       присутствует опционально, если открытый ключ
   POPOSigningKey   из crm[0].certReq.certTemplate является ключом подписи
  -- в этом обмене МОЖЕТ потребоваться подтверждение владения (см. Приложение D.3)
crm[0].certReq.     присутствует опционально
   controls.archiveOptions
  -- конечный элемент МОЖЕТ запросить архивирование локально сгенерированного секретного
  -- ключа
crm[0].certReq.     присутствует опционально
   controls.publicationInfo
  -- конечный элемент МОЖЕТ запросить публикацию полученного в результате сертификата
crm[1].certReq      Фиксированное значение 1
   certReqId
  -- индекс шаблона в сообщении
crm[1].certReq      присутствует
   certTemplate
   -- НЕДОПУСТИМО включение реальных битов открытого ключа, других ограничений нет
   -- (например, имена не обязаны совпадать с именами в crm[0]).
   -- Отметим, что поле subjectPublicKeyInfo МОЖЕТ присутствовать и содержать
   -- AlgorithmIdentifier, за которым следует  BIT STRING нулевого размера для
   -- subjectPublicKey, если желательно информировать CA/RA об алгоритме и параметрах,
   -- которые являются предпочтительными для генерируемой пары ключей.
crm[1].certReq.     присутствует [идентификатор объекта ДОЛЖЕН быть PROT_ENC_ALG]
   controls.protocolEncrKey
  -- если централизованная генерация ключей поддерживается данным агентом CA, этот
  -- краткосрочный асимметричный ключ шифрования (генерируемый конечным элементом) будет
  -- использоваться CA для шифрования (симметричного ключа, используемого для шифрования)
  -- секретного ключа, генерируемого CA от имени конечного элемента
crm[1].certReq.     присутствует опционально
```

```
      controls.archiveOptions
   crm[1].certReq.       присутствует опционально
      controls.publicationInfo
   protection            присутствует
     -- биты, рассчитанные с использованием MSG_MAC_ALG
```

Отклик при инициализации - ip

```
   Поле                  Значение
   sender                имя CA
     -- имя агента CA, создавшего сообщение
   messageTime           присутствует
     -- время создания сообщения агентом CA
   protectionAlg         MS_MAC_ALG
     -- для этого отклика разрешена только MAC-защита
   senderKID             referenceNum
     -- ссылочный номер, который CA ранее эмиттировал для конечного элемента (вместе с ключом
     -- MACing)
   transactionID         присутствует
     -- значение из соответствующего сообщения ir
   senderNonce           присутствует
     -- 128 (псевдо-)случайных битов
   recipNonce            присутствует
     -- значение из senderNonce соответствующего сообщения ir
   freeText              любое допустимое значение
   body                  ip (CertRepMessage)
                         содержит в точности один отклик для каждого запроса

     -- PKI (CA) отвечает на один или два запроса подобающим образом. crc[0] обозначает
     -- первое (присутствует всегда); crc[1] - второе (присутствует лишь в случаях, когда
     -- сообщение ir содержит два запроса и CA поддерживает централизованную генерацию ключей)
   crc[0].               фиксированное значение 0
      certReqId
     -- ДОЛЖНО содержать отклик на первый запрос в соответствующем сообщении ir

   crc[0].status.        присутствует, разрешены позитивные значения
      status                "accepted", "grantedWithMods"
                         и негативное значение "rejection"
   crc[0].status.        присутствует тогда и только тогда, когда
      failInfo           crc[0].status.status = "rejection"
   crc[0].               присутствует  тогда и только тогда, когда
      certifiedKeyPair   crc[0].status.status имеет значение "accepted" или "grantedWithMods"
   certificate           присутствует, если открытый ключ конечного элемента не является ключом
                         шифрования и POP не выполняется в этом обмене сообщениями
   encryptedCert         присутствует  тогда и только тогда, когда открытый ключ является
                         ключом шифрования и POP осуществляется в этом обмене сообщениями
   publicationInfo       присутствует опционально

     -- показывает, где будет публиковаться сертификат (включается по усмотрению CA)

   crc[1].               фиксированное значение 1
      certReqId
     -- ДОЛЖНО содержать отклик на второй запрос в соответствующем сообщении ir
   crc[1].status.        присутствует, разрешены позитивные значения
      status                "accepted", "grantedWithMods"
                         и негативное значение "rejection"
   crc[1].status.        присутствует тогда и только тогда, когда
      failInfo           crc[0].status.status = "rejection"
   crc[1].               присутствует  тогда и только тогда, когда
      certifiedKeyPair   crc[0].status.status имеет значение "accepted" или "grantedWithMods"
   certificate           присутствует

   privateKey            присутствует
     -- см. Приложение Приложение C. Разъяснение поведения для запросов
   publicationInfo       присутствует опционально
     -- показывает, где будет публиковаться сертификат (включается по усмотрению CA)

   protection            присутствует
     -- биты, рассчитанные с использованием MSG_MAC_ALG
   extraCerts            присутствует опционально
     -- агент CA МОЖЕТ обеспечить конечному элементу дополнительные сертификаты
```

Подтверждение сертификата - certConf

```
   Поле                  Значение
   sender                присутствует
     -- как для сообщений ir
   recipient             имя CA
     -- имя агента CA, у которого запрошен выпуск сертификата
   transactionID         присутствует
     -- значение из соответствующих сообщений ir и ip
   senderNonce           присутствует
     -- 128 (псевдо-)случайных битов
   recipNonce            присутствует
     -- значение из senderNonce соответствующего сообщения ip
   protectionAlg         MSG_MAC_ALG
```

```
    -- для этого сообщения разрешена только MAC-защита. Значение MAC создаётся на базе
    -- начального ключа идентификации, совместно используемого EE и CA.

  senderKID             referenceNum
    -- ссылочный номер, который CA ранее эмиттировал для конечного элемента (вместе с ключом
    -- MACing)

  body                  certConf
    -- см. параграф 5.3.18. Содержимое подтверждения сертификата, где описано содержимое
    -- полей certConf. Примечание: две структуры CertStatus требуются в случаях, когда
    -- передаются оба сертификата (для шифрования и подписи).

  protection            присутствует
    -- биты, рассчитанные с использованием MSG_MAC_ALG
```

Подтверждение - PKIConf

```
  Поле                  Значение
  sender                присутствует
    -- как для сообщений ip
  recipient             присутствует
    -- имя отправителя из certConf
  transactionID         присутствует
    -- значение из сообщения certConf
  senderNonce           присутствует
    -- 128 (псевдо-)случайных битов
  recipNonce            присутствует
    -- значение из поля senderNonce сообщения certConf
  protectionAlg         MSG_MAC_ALG
    -- для этого сообщения разрешена только MAC-защита
  senderKID             referenceNum
  body                  PKIConf
  protection            присутствует
    -- биты, рассчитанные с использованием MSG_MAC_ALG
```

D.5. Запрос сертификата

(Инициализированный) конечный элемент запрашивает сертификат у агента CA (с любой целью). Когда CA отвечает сообщением, содержащим сертификат, конечный элемент отвечает на это сообщение подтверждением сертификата. CA тогда передаёт сообщение PKIConfirm для завершения транзакции. Все сообщения заверяются.

Профиль для этого обмена идентичен профилю, описанному в Приложении D.4 с некоторыми исключениями:

- **следует** включать имя отправителя;
- **должно** поддерживаться поле protectionAlg в MSG_SIG_ALG (**может** поддерживаться также MSG_MAC_ALG) для сообщений запроса, отклика certConfirm и PKIConfirm;
- поля senderKID и recipKID присутствуют только в тех случаях, когда они требуются для верификации сообщения;
- тело сообщения представляет собой cr или cp;
- тело сообщения может содержать одну или две структуры CertReqMsg — любая из них может использоваться для запроса сертификации локально сгенерированного открытого ключа или сгенерированного централизованно открытого ключа (т. е., требование по размещению этих структур, приведённое в Приложении D.4, в данном случае снимается);
- биты защиты рассчитываются в соответствии с полем protectionAlg.

D.6. Запрос обновления ключа

(Инициализированный) конечный элемент запрашивает сертификат у агента CA (для обновления ключевой пары и/или соответствующего сертификата, которым он уже владеет). Когда CA отвечает сообщением, содержащим сертификат, конечный элемент отвечает на это сообщение подтверждением сертификата. CA тогда передаёт сообщение PKIConfirm для завершения транзакции. Все сообщения заверяются.

Профиль для этого обмена идентичен профилю, описанному в Приложении D.4 с некоторыми исключениями:

1. **следует** включать имя отправителя;
2. **должно** поддерживаться поле protectionAlg в MSG_SIG_ALG (**может** поддерживаться также MSG_MAC_ALG) для сообщений запроса, отклика certConfirm и PKIConfirm;
3. поля senderKID и recipKID присутствуют только в тех случаях, когда они требуются для верификации сообщения;
4. тело представляет собой kur или kup;
5. тело сообщения может содержать одну или две структуры CertReqMsg — любая из них может использоваться для запроса сертификации локально сгенерированного открытого ключа или сгенерированного централизованно открытого ключа (т. е., требование по размещению этих структур, приведённое в Приложении D.4, в данном случае снимается);
6. биты защиты рассчитываются в соответствии с полем protectionAlg;
7. **следует** использовать regCtrl OldCertId (пока для отправителя и получателя с помощью не задаваемых этой спецификацией мер не становится очевидной ненужность этого).

Приложение E. Профили управляющих сообщений PKI (опция).

В этом приложении приведены подробные профили для тех сообщений PKIMessage, которые **могут** поддерживаться реализациями (в дополнение к тем сообщениям, которые **должны** поддерживаться — см. раздел 6 и Приложение D).

Представлены профили сообщений PKIMessage для следующих операций управления PKI:

- обновление ключа корневого CA;
- информационный запрос/отклик;
- запрос/отклик при кросс-сертификации (1-сторонний)
- инициализация по основному каналу с использованием внешнего сертификата тождественности.

Будущие версии этого документа могут быть расширены путём добавления профилей для перечисленных ниже операций (вместе с другими нужными операциями):

- запрос отзыва;
- публикация сертификата;
- публикация CRL.

E.1. Общие правила интерпретации профилей

Идентично Приложению D.1.

E.2. Алгоритм использования профиля

Идентично Приложению D.2.

E.3. Самоподписанные сертификаты

Профиль показывает, как структура Certificate может быть «самоподписана». Такие структуры используются для распространения открытых ключей агентов CA. Процесс может выполняться одним из трёх способов (см. параграф 4.4, где дано описание этих структур.

Тип	*Назначение*
`newWithNew`	Корректный «самоподписанный» сертификат; содержащийся открытый ключ **должен** быть пригоден для верификации подписи (хотя при этом обеспечивается только проверка целостность и не проверяется идентификация).
`oldWithNew`	Прежний открытый ключ корневого CA с новым секретным ключом.
`newWithOld`	Новый открытый ключ корневого CA с прежним секретным ключом.

Такие сертификаты (включая подходящие расширения) должны содержать «разумные» значения для всех полей. Например, поле subjectAltName (при его наличии) **должно** быть идентично полю issuerAltName, а поле keyIdentifier (при его наличии) должно содержать приемлемое значение и т. п.

E.4. Обновление ключа корневого CA

Корневой агент CA обновляет свою пару ключей. В этом случае агент создаёт анонс обновления ключей, который может быть доступен (с использованием того или иного транспортного механизма) соответствующим конечным элементам. От конечных элементов подтверждающих сообщений **не требуется**.

Сообщение ckuann

Поле	*Значение*	*Комментарий*
`sender`	`имя CA имя CA`	
`body`	`ckuann(CAKeyUpdAnnContent)`	
`oldWithNew`	присутствует	см. Приложение E.3
`newWithOld`	присутствует	см. Приложение E.3
`newWithNew`	присутствует	см. Приложение E.3
`extraCerts`	присутствует опционально	может использоваться для «публикации» сертификатов (например, сертификатов, подписанных с использованием нового секретного ключа

E.5. Информационный запрос/отклик PKI

Конечный элемент передаёт сообщение общего назначения PKI, запрашивая детальную информацию, которая может потребоваться для последующих операций управления PKI. Агент RA/CA даёт отклик общего назначения. Если отклик генерирует агент RA, это будет просто пересылка соответствующего сообщения, полученного ранее от CA, с возможным добавлением сертификатов в поля extraCerts сообщения PKIMessage. Подтверждения от конечного элемента **не требуется**.

Поток сообщений

Этап	*Конечный элемент*		*PKI*
1	`Форматирование genm`		
2		`-> genm ->`	
3			`Обработка genm`
4			`Создание genp`
5		`<- genp <-`	
6	`Обработка genp`		

genM

`Поле`	`Значение`
`recipient`	`имя CA`

```
    -- имя агента CA, как оно указано в расширениях issuerAltName или полях
    -- issuer сертификатов
  protectionAlg       MSG_MAC_ALG или MSG_SIG_ALG
    -- любой заверенный алгоритм защиты
  SenderKID           присутствует при необходимости
    -- должно присутствовать, если требуется для верификации защиты сообщения
  freeText            любое корректное значение
  body                genr (GenReqContent)
  GenMsgContent       пустая SEQUENCE
    -- вся запрошенная информация, имеющая отношение к делу
  protection          присутствует
    -- биты, рассчитанные с использованием MSG_MAC_ALG или MSG_SIG_ALG
```

genP

```
  Поле                Значение
  sender              имя CA
    -- имя агента CA, создавшего сообщение
  protectionAlg       MSG_MAC_ALG или MSG_SIG_ALG
    -- любой заверенный алгоритм защиты
  senderKID           присутствует при необходимости
    -- должно присутствовать, если требуется для верификации защиты сообщения
  body                genp (GenRepContent)
  CAProtEncCert       присутствует (идентификатор объекта одного из PROT_ENC_ALG), с
                      соответствующим значением
    -- будет использоваться, если конечному элементу необходимо зашифровать информацию для
    -- CA (например, секретный ключ для восстановления)

  SignKeyPairTypes    присутствует с соответствующим значением
    -- набор идентификаторов алгоритмов защиты, которые данный CA будет сертифицировать
    -- для открытых ключей субъекта
  EncKeyPairTypes     присутствует с соответствующим значением
    -- набор алгоритмов шифрования/согласования ключей, которые данный CA будет
    -- сертифицировать для открытых ключей субъекта
  PreferredSymmAlg    присутствует (идентификатор объекта одного из PROT_ENC_ALG), с
                      соответствующим значением
    -- симметричный алгоритм, который данный CA предполагает использовать в последующих
    -- сообщениях PKI (для шифрования)
  CAKeyUpdateInfo     опционально присутствует с соответствующим значением
    -- CA МОЖЕТ предоставлять информацию о соответствующей паре ключей корневого CA
    -- с использованием этого поля (отметим, что это не предполагает того, что отвечающий
    -- агент CA является корневым CA, о котором идёт речь)
  CurrentCRL          опционально присутствует с соответствующим значением
    -- CA МОЖЕТ предоставлять копию полного списка CRL (т. е., максимально заполненную)
  protection          присутствует
    -- биты, рассчитанные с использованием MSG_MAC_ALG или MSG_SIG_ALG
  extraCerts          опционально присутствует
    -- может использоваться для передачи некоторых сертификатов конечному элементу;
    -- RA МОЖЕТ добавить сюда свой сертификат.
```

E.6. Запрос/отклик кросс-сертификации (односторонний)

Создание одного кросс-сертификата (т. е., не двух в один приём). Запрашивающий агент CA **может** выбрать, кто будет отвечать за публикацию кросс-сертификата, выпущенного отвечающим CA, с помощью элемента PKIPublicationInfo.

Предпосылки.

1. отвечающий CA может проверить источник запроса (возможно, это потребует «автономных» мер) перед обработкой запроса;
2. отвечающий CA может заверить подлинность источника запроса (возможно, это потребует «автономных» мер) перед обработкой запроса.

Использование подтверждения сертификата и подтверждения соответствующего сервера определяется полем generalInfo в заголовке PKIHeader (параграф 5.1.1). Приведённый профиль не требует каких-либо подтверждений.

Поток сообщений

```
Этап Запрашивающий CA            Отвечающий CA
  1  Форматирование ccr
  2                      -> ccr ->
  3                                Обработка ccr
  4                                Создание ccp
  5                      <- ccp <-
  6  Обработка ccp
```

ccr

```
  Поле                Значение
  sender              имя запрашивающего CA
    -- имя агента CA, создавшего сообщение
  recipient           имя отвечающего CA
    -- имя агента CA, у которого запрашивается создание кросс-сертификата
  messageTime         время создания сообщения
    -- текущее время запрашивающего CA
  protectionAlg       MSG_SIG_ALG
    -- для этого запроса допустима только защита цифровой подписью
  senderKID           присутствует при необходимости
    -- должно присутствовать, если требуется верификация защиты сообщения
```

```
recipKID              присутствует при необходимости
  -- должно присутствовать, если требуется верификация защиты сообщения
transactionID         присутствует
  -- зависящее от реализации значение, понятное запрашивающему CA (если такой идентификатор
  -- уже используется на отвечающем CA, им ДОЛЖНО генерироваться сообщение об отказе)
senderNonce           присутствует
  -- 128 (псевдо-)случайных битов
freeText              любое допустимое значение
body                  ccr (CertReqMessages)
                      разрешается только одно поле CertReqMsg
  -- если требуется множество кросс-сертификатов, они ДОЛЖНЫ помещаться в отдельные
  -- сообщения PKIMessage
certTemplate          присутствует
  -- Детали приведены ниже
version               v1 или v3
  -- НАСТОЯТЕЛЬНО РЕКОМЕНДУЕТСЯ v3
signingAlg            присутствует
  -- запрашивающий CA должен знать заранее с каким алгоритмом он хочет получить
  -- подпись в сертификате

subject               присутствует
  -- может принимать значение NULL-DN только при наличии расширения subjectAltNames
validity              присутствует
  -- ДОЛЖНО быть задано полностью (т. е., оба поля должны присутствовать)
issuer                присутствует
  -- может принимать значение NULL-DN только при наличии расширения issuerAltNames
publicKey             присутствует
  -- ключ, который будет сертифицирован (должен быть ключом алгоритма подписи)
extensions            присутствует опционально
  -- запрашивающий CA должен предложить значения для всех расширений, которые требуются
  -- в кросс-сертификате
POPOSigningKey        присутствует
  -- см. Приложение D.3. Подтверждение владения профилем
protection            присутствует
  -- биты, рассчитанные с использованием MSG_SIG_ALG
extraCerts            присутствует опционально
  -- МОЖЕТ содержать любые дополнительные сертификаты, которые запрашивающая сторона
  -- желает включить
```

ccp

```
Поле                  Значение
sender                имя отвечающего CA
  -- имя агента CA, создавшего сообщение
recipient             имя запрашивающего CA
  -- имя агента CA, который запросил выпуск сертификата
messageTime           время создания сообщения
  -- текущее время отвечающего CA
protectionAlg         MSG_SIG_ALG
  -- для защиты этого сообщения может использоваться только цифровая подпись
senderKID             присутствует при необходимости
  -- должно присутствовать, если требуется верификация защиты сообщения
recipKID              присутствует при необходимости
transactionID         присутствует
  -- значение из соответствующего сообщения ccr
senderNonce           присутствует
  -- 128 (псевдо-)случайных битов
recipNonce            присутствует
-- senderNonce из соответствующего сообщения ccr
freeText              любое допустимое значение
body                  ccp (CertRepMessage)
                      допустимо только одно поле CertResponse
  -- если требуется множество кросс-сертификатов, они ДОЛЖНЫ помещаться в отдельные
  -- сообщения PKIMessage
response              присутствует
status                присутствует

PKIStatusInfo.status  присутствует
  -- если PKIStatusInfo.status имеет значение accepted или grantedWithMods, поле
  -- certifiedKeyPair ДОЛЖНО присутствовать, а failInfo ДОЛЖНО отсутствовать

failInfo              присутствует в зависимости от PKIStatusInfo.status
  -- если PKIStatusInfo.status = rejection, поле certifiedKeyPair ДОЛЖНО отсутствовать,
  -- а failInfo ДОЛЖНО присутствовать и содержать подходящие установки битов

certifiedKeyPair      присутствует в зависимости от PKIStatusInfo.status
certificate           присутствует в зависимости от  certifiedKeyPair

  -- содержимое актуального сертификата должно проверяться запрашивающим CA до публикации
protection            присутствует
  -- биты, рассчитанные с использованием MSG_SIG_ALG
extraCerts            присутствует опционально
  -- МОЖЕТ содержать любые дополнительные сертификаты, которые отвечающая сторона
  -- желает включить
```

E.7. Инициализация по основному каналу с использованием внешнего сертификата тождественности

(Неинициализированный) конечный элемент желает инициализировать себя в PKI с использованием агента CA-1. Он использует для заверения существующий сертификат, эмиттированный агентом CA-X. Между CA-1 и CA-X уже должны быть установлены доверительные отношения, поэтому CA-1 может подтвердить сертификат тождественности EE, подписанный агентом CA-X. Кроме того, требуется организация неких механизмов в среде персональной защиты (PSE[1]) элемента EE, позволяющие заверить и проверить сообщения PKIMessage, подписанные CA-1 (например, PSE может включать сертификат, выпущенный для открытого ключа CA-1, подписанный другим CA, которому EE доверяет на основе автономных механизмов заверения).

EE передаёт инициализационный запрос для начала транзакции. Когда CA-1 ответит сообщением, содержащим новый сертификат, конечный элемент ответит на это сообщение подтверждением сертификата. CA-1 ответит сообщением PKIConfirm для завершения транзакции. Все сообщения подписываются (сообщения EE подписываются с использованием секретного ключа, который соответствует открытому ключу внешнего сертификата тождественности; сообщения CA-1 подписываются с использованием секретного ключа, соответствующего открытому ключу в сертификате, который связан с доверенной привязкой в среде PSE конечного элемента).

Профиль для этого обмена идентичен профилю, описанному в приложении D.4, с некоторыми исключениями:

- EE и CA-1 не имеют разделяемого симметричного ключа MACing (т. е., здесь не организуется разделяемая с помощью автономных мер между элементами секретная информация);
- имя отправителя в ir **должно** присутствовать (и совпадать с именем субъекта во внешнем сертификате тождественности);
- поле protectionAlg MSG_SIG_ALG **должно** использоваться во всех сообщениях;
- внешний сертификат тождественности **должен** передаваться в поле extraCerts сообщения ir;
- senderKID и recipKID не используются;
- телом сообщения является ir или ip;
- биты защиты рассчитываются в соответствии с полем protectionAlg.

Приложение F. Компилируемые определения ASN.1

```
PKIXCMP {iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
          pkix(7) id-mod(0) id-mod-cmp2000(16)}

     DEFINITIONS EXPLICIT TAGS ::=

     BEGIN
     -- Экспортировать все --
     IMPORTS
         Certificate, CertificateList, Extensions, AlgorithmIdentifier,
         UTF8String -- если требуется; в противном случае «закомментировать»
                FROM PKIX1Explicit88 {iso(1) identified-organization(3) dod(6) internet(1)
                security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-88(1)}

         GeneralName, KeyIdentifier
                FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1)
                security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-88(2)}

         CertTemplate, PKIPublicationInfo, EncryptedValue, CertId, CertReqMessages
                FROM PKIXCRMF-2005 {iso(1) identified-organization(3) dod(6) internet(1)
                security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005(36)}

         -- см. также характеристики поведения CRMF в Приложении C к данной спецификации

         CertificationRequest
                FROM PKCS-10 {iso(1) member-body(2) us(840) rsadsi(113549)
                              pkcs(1) pkcs-10(10) modules(1) pkcs-10(1)}

         -- (задано в RFC 2986 с синтаксисом 1993 ASN.1 и ЯВНЫМИ тегами); в дополнение к этому
         -- разработчики могут напрямую включить в этот модуль синтаксис [PKCS10]
         ;

   -- остальная часть модуля содержит локально определяемые идентификаторы OID и конструкции

      CMPCertificate ::= CHOICE {
         x509v3PKCert        Certificate
      }
   -- Этот синтаксис, будучи совместимым на уровне битового представления сигналов в проводе,
   -- с определением Certificate в стандарте X.509, обеспечивает возможность введения новых
   -- типов сертификатов (таких, как сертификаты атрибутов X.509, сертификаты WAP WTLS и др.)
   -- в рамках данного протокола управления сертификатами, не требуя ничего для поддержки
   -- такой общности. Реализации, которые не видят необходимости поддерживать сертификаты
   -- других типов, МОГУТ, при желании, «закомментировать» приведённую выше структуру и
   -- «раскомментировать» приведённую ниже структуру до компиляции модуля ASN.1 (отметим,
   -- что это не оказывает влияния на взаимодействие с реализациями, не делающими так)

   -- CMPCertificate ::= Certificate
```

[1]Personal Security Environment.

```
 PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL
}

PKIMessages ::= SEQUENCE SIZE (1..MAX) OF PKIMessage

PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2) },
    sender              GeneralName,
    -- идентифицирует отправителя
    recipient           GeneralName,
    -- идентифицирует адресата
    messageTime     [0] GeneralizedTime         OPTIONAL,
    -- время создания сообщения (используется, когда отправитель предполагает, что
    -- транспорт будет «подходящим» (т. е., значение будет иметь смысл и при получении)
    protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
    -- алгоритм, используемый для расчёта битов защиты
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    -- идентификация конкретных ключей, используемых для защиты
    transactionID   [4] OCTET STRING            OPTIONAL,
    -- идентифицирует транзакцию (т. е., будет совпадать в соответствующих сообщениях
    -- запроса, отклика, certConf и PKIConf
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    -- временные маркеры, служащие для защиты от повторного использования; значение
    -- senderNonce помещается создателем сообщения, а recipNonce является маркером,
    -- ранее помещённым в соответствующее сообщение адресатом данного сообщения
    freeText        [7] PKIFreeText             OPTIONAL,
    -- может использоваться для передачи контекстно-зависимых инструкций
    -- (предназначено для чтения людьми)
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue     OPTIONAL
    -- может использоваться для передачи контекстно-зависимой информации
    -- (предназначено для чтения людьми)
}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
    -- текст представляется в форме UTF-8 String [RFC3629] (каждая строка UTF8String
    -- МОЖЕТ включать язык тегов [RFC3066] для индикации языка содержащегося в
    -- поле текста; подробное описание содержится в [RFC2482])

PKIBody ::= CHOICE {       -- специфичные для сообщения элементы тела сообщения
    ir       [0]  CertReqMessages,        --Запрос инициализации
    ip       [1]  CertRepMessage,         --Отклик при инициализации
    cr       [2]  CertReqMessages,        --Запрос сертификации
    cp       [3]  CertRepMessage,         --Отклик при сертификации
    p10cr    [4]  CertificationRequest,   --импортировано из [PKCS10]
    popdecc  [5]  POPODecKeyChallContent, --Вызов pop
    popdecr  [6]  POPODecKeyRespContent,  --Отклик при pop
    kur      [7]  CertReqMessages,        --Запрос на обновление ключа
    kup      [8]  CertRepMessage,         --Отклик при обновлении ключа
    krr      [9]  CertReqMessages,        --Запрос на восстановление ключа
    krp      [10] KeyRecRepContent,       --Отклик при восстановлении ключа
    rr       [11] RevReqContent,          --Запрос на отзыв
    rp       [12] RevRepContent,          --Отклик при отзыве
    ccr      [13] CertReqMessages,        --Запрос кросс-сертификации
    ccp      [14] CertRepMessage,         --Отклик при кросс-сертификации
    ckuann   [15] CAKeyUpdAnnContent,     --Анонс обновления ключа CA
    cann     [16] CertAnnContent,         --Анонс сертификата
    rann     [17] RevAnnContent,          --Анонс отзыва
    crlann   [18] CRLAnnContent,          --Анонс CRL
    pkiconf  [19] PKIConfirmContent,      --Подтверждение
    nested   [20] NestedMessageContent,   --Вложенное сообщение
    genm     [21] GenMsgContent,          --Сообщение общего назначения
    genp     [22] GenRepContent,          --Отклик общего назначения
    error    [23] ErrorMsgContent,        --Сообщение об ошибке
    certConf [24] CertConfirmContent,     --Подтверждение сертификата
    pollReq  [25] PollReqContent,         --Запрос опроса
    pollRep  [26] PollRepContent          --Отклик при опросе
}

PKIProtection ::= BIT STRING

ProtectedPart ::= SEQUENCE {
    header    PKIHeader,
    body      PKIBody
}

id-PasswordBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
    salt                OCTET STRING,
    -- Примечание: реализация МОЖЕТ ограничивать допустимый размер строки для снижения
```

```
        -- риска атак на службы
        owf                 AlgorithmIdentifier,
        -- AlgId для необратимой функции (рекомендуется SHA-1)
        iterationCount      INTEGER,
        -- число итераций OWF
        -- Примечание: реализация МОЖЕТ ограничивать допустимый размер строки для снижения
        -- риска атак на службы
        mac                 AlgorithmIdentifier
        -- MAC AlgId (например, DES-MAC, Triple-DES-MAC [PKCS11] или HMAC [RFC2104, RFC2202])
    }

    id-DHBasedMac OBJECT IDENTIFIER ::= {1 2 840 113533 7 66 30}
    DHBMParameter ::= SEQUENCE {
        owf                 AlgorithmIdentifier,
        -- для необратимой функции (рекомендуется SHA-1)
        mac                 AlgorithmIdentifier
        -- MAC AlgId (например, DES-MAC, Triple-DES-MAC [PKCS11] или HMAC [RFC2104, RFC2202])
    }


    NestedMessageContent ::= PKIMessages

    PKIStatus ::= INTEGER {
        accepted                (0),
        -- получено именно то, что запрашивалось
        grantedWithMods         (1),
        -- получено что-то похожее на запрошенную информацию; запрашивающая сторона
        -- принимает на себя ответственность за нахождение различий
        rejection               (2),
        -- не получено, дополнительная информация в сообщении
        waiting                 (3),
        -- тело запроса ещё не обработано, отклик будет позднее (примечание: при корректной
        -- обработке такого отклика МОГУТ использоваться опросные сообщения «запрос-отклик»,
        -- описанные в параграфе 5.3.22; другим вариантом является опрос нижележащего
        -- транспортного уровня, который МОЖЕТ оказаться полезным)
        revocationWarning       (4),
        -- сообщение содержит предупреждение о неминуемости отзыва
        revocationNotification (5),
        -- уведомление о произошедшем отзыве
        keyUpdateWarning        (6)
        -- обновление уже выполнено для oldCertId, заданного в CertReqMsg
    }

    PKIFailureInfo ::= BIT STRING {
    -- Поскольку возможно множество причин отказов, в будущем могут быть добавлены новые коды
        badAlg              (0),
        -- нераспознанный или неподдерживаемый идентификатор алгоритма
        badMessageCheck     (1),
        -- отказ при проверке целостности (например, подпись не соответствует)
        badRequest          (2),
        -- транзакция не разрешена или не поддерживается
        badTime             (3),
        -- значение messageTime отличается от системного времени более, чем позволяет
        -- локальная политика
        badCertId           (4),
        -- не найдено сертификата, соответствующего представленным критериям
        badDataFormat       (5),
        -- данные представлены в некорректном формате
        wrongAuthority      (6),
        -- агентство, указанное в запросе отличается от создавшего маркер отклика
        incorrectData       (7),
        -- данные запрашивающей стороны некорректны (для нотариальных услуг)
        missingTimeStamp    (8),
        -- временная метка отсутствует, но требуется (политикой)
        badPOP              (9),
        -- отказ при проверке pop
        certRevoked         (10),
           -- сертификат уже отозван
        certConfirmed       (11),
           -- сертификат уже подтверждён
        wrongIntegrity      (12),
           -- некорректный контроль целостности — на базе пароля взамен подписи или наоборот
        badRecipientNonce   (13),
           -- некорректный временный маркер получателя (значение отсутствует или некорректно)
        timeNotAvailable    (14),
           -- данные о времени TSA недоступны
        unacceptedPolicy    (15),
           -- запрошенная политика TSA не поддерживается TSA
        unacceptedExtension (16),
           -- запрошенное расширение не поддерживается TSA
        addInfoNotAvailable (17),
           -- запрошенная дополнительная информация непонятна или недоступна
        badSenderNonce      (18),
           -- некорректный временный маркер отправителя (значение отсутствует или некорректно)
        badCertTemplate     (19),
           -- некорректный шаблон сертификата или отсутствует обязательная информация
```

```
        signerNotTrusted    (20),
           -- подписавшая сторона неизвестна или не пользуется доверием
        transactionIdInUse  (21),
           -- идентификатор транзакции уже используется
        unsupportedVersion  (22),
           -- версия сообщения не поддерживается
        notAuthorized       (23),
           -- отправитель не уполномочен выполнять предшествующий запрос или действие
        systemUnavail       (24),
        -- запрос не может быть обработан по причине системной недоступности
        systemFailure       (25),
        -- запрос не может быть обработан по причине системно отказа
        duplicateCertReq    (26)
        -- сертификат не может быть выпущен, поскольку уже существует дубликат сертификата
    }

    PKIStatusInfo ::= SEQUENCE {
        status        PKIStatus,
        statusString  PKIFreeText     OPTIONAL,
        failInfo      PKIFailureInfo  OPTIONAL
    }

    OOBCert ::= CMPCertificate

    OOBCertHash ::= SEQUENCE {
        hashAlg     [0] AlgorithmIdentifier     OPTIONAL,
        certId      [1] CertId                  OPTIONAL,
        hashVal         BIT STRING
        -- hashVal рассчитывается для DER-представления самоподписанного сертификата с
        -- идентификатором certID.
    }

    POPODecKeyChallContent ::= SEQUENCE OF Challenge
    -- Используется один запрос Challenge на запрос сертификации ключа шифрования (в том же
    -- порядке, как эти запросы указаны в сообщениях CertReqMessage).

    Challenge ::= SEQUENCE {
        owf                 AlgorithmIdentifier  OPTIONAL,
        -- ДОЛЖНО присутствовать в первом Challenge; МОЖЕТ быть опущено в последующих
        -- Challenge в POPODecKeyChallContent (если опущено, будет использоваться owf из
        -- непосредственно предшествующего Challenge).

        witness             OCTET STRING,
        -- результат применения необратимой функции (owf) к случайному числу INTEGER, A.
        -- (отметим, что для каждого Challenge ДОЛЖНЫ использоваться разные значения INTEGER)
        challenge           OCTET STRING
        -- зашифрованное (с использованием открытого ключа, для которого запрашивается
        -- сертификат) значение Rand, где Rand задано, как показано ниже
        --   Rand ::= SEQUENCE {
        --      int      INTEGER,
        --       - случайное значение INTEGER A (см. выше)
        --      sender   GeneralName
        --       - имя отправителя (как в заголовке PKIHeader)
        --   }
    }

    POPODecKeyRespContent ::= SEQUENCE OF INTEGER
    -- Одно значение INTEGER на запрос сертификации ключа шифрования (в том же порядке,
    -- который используется в сообщениях CertReqMessage). Восстановленное значение INTEGER A
    -- (см. выше) возвращается отправителю соответствующего Challenge.

    CertRepMessage ::= SEQUENCE {
        caPubs       [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL,
        response         SEQUENCE OF CertResponse
    }

    CertResponse ::= SEQUENCE {
        certReqId           INTEGER,
        -- для сопоставления этого отклика с соответствующим запросом (значение -1
        -- используется в том случае, когда значение certReqId не задано в запросе)

        status              PKIStatusInfo,
        certifiedKeyPair    CertifiedKeyPair    OPTIONAL,
        rspInfo             OCTET STRING        OPTIONAL
        -- аналогично строке id-regInfo-utf8Pairs, определённой для regInfo в CertReqMsg
        -- [CRMF]
    }

    CertifiedKeyPair ::= SEQUENCE {
        certOrEncCert       CertOrEncCert,
        privateKey      [0] EncryptedValue      OPTIONAL,
        -- см. комментарии по представлению в [CRMF]
        publicationInfo [1] PKIPublicationInfo  OPTIONAL
    }

    CertOrEncCert ::= CHOICE {
```

```
    certificate     [0] CMPCertificate,
    encryptedCert   [1] EncryptedValue
}

KeyRecRepContent ::= SEQUENCE {
    status                  PKIStatusInfo,
    newSigCert          [0] CMPCertificate OPTIONAL,
    caCerts             [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate OPTIONAL,
    keyPairHist         [2] SEQUENCE SIZE (1..MAX) OF CertifiedKeyPair OPTIONAL
}

RevReqContent ::= SEQUENCE OF RevDetails

RevDetails ::= SEQUENCE {
    certDetails         CertTemplate,
    -- позволяет запрашивающему подробно описать сертификат, для которого запрашивается
    -- отзыв (например, для случаев, когда значение serialNumber недоступно)
    crlEntryDetails     Extensions       OPTIONAL
    -- запрошенные расширения crlEntryExtensions
}

RevRepContent ::= SEQUENCE {
    status       SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
    -- в том же порядке, как передано в RevReqContent
    revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId     OPTIONAL,
    -- значения ID для каждого запрошенного отзыва (в том же порядке, как статус)
    crls     [1] SEQUENCE SIZE (1..MAX) OF CertificateList    OPTIONAL
    -- результирующие списки CRL (может быть несколько)
}

CAKeyUpdAnnContent ::= SEQUENCE {
    oldWithNew   CMPCertificate, -- старый открытый ключ, подписанный новым секретным
    newWithOld   CMPCertificate, -- новый открытый ключ, подписанный старым секретным
    newWithNew   CMPCertificate  -- новый открытый ключ, подписанный новым секретным
}

CertAnnContent ::= CMPCertificate

RevAnnContent ::= SEQUENCE {
    status              PKIStatus,
    certId              CertId,
    willBeRevokedAt     GeneralizedTime,
    badSinceDate        GeneralizedTime,
    crlDetails          Extensions  OPTIONAL
    -- дополнительные детали CRL (например, номер, причина, местоположение и т. п.)
}

CRLAnnContent ::= SEQUENCE OF CertificateList

CertConfirmContent ::= SEQUENCE OF CertStatus

CertStatus ::= SEQUENCE {
   certHash    OCTET STRING,
   -- хэш-сумма для сертификата, вычисленная с использованием того же алгоритма,
   -- который применяется для создания и проверки сертификата подписи
   certReqId   INTEGER,
   -- для сопоставления данного подтверждения с соответствующим «запросом-откликом»
   statusInfo  PKIStatusInfo OPTIONAL
}

PKIConfirmContent ::= NULL

InfoTypeAndValue ::= SEQUENCE {
    infoType               OBJECT IDENTIFIER,
    infoValue              ANY DEFINED BY infoType  OPTIONAL
}
-- Пример содержимого InfoTypeAndValue включает, но не ограничивается приведённым ниже
-- («раскомментируйте» в этом модуле ASN.1 и используйте подходящее для вашей среды:
--
--   id-it-caProtEncCert    OBJECT IDENTIFIER ::= {id-it 1}
--      CAProtEncCertValue      ::= CMPCertificate
--   id-it-signKeyPairTypes OBJECT IDENTIFIER ::= {id-it 2}
--      SignKeyPairTypesValue   ::= SEQUENCE OF AlgorithmIdentifier
--   id-it-encKeyPairTypes  OBJECT IDENTIFIER ::= {id-it 3}
--      EncKeyPairTypesValue    ::= SEQUENCE OF AlgorithmIdentifier
--   id-it-preferredSymmAlg OBJECT IDENTIFIER ::= {id-it 4}
--      PreferredSymmAlgValue   ::= AlgorithmIdentifier
--   id-it-caKeyUpdateInfo  OBJECT IDENTIFIER ::= {id-it 5}
--      CAKeyUpdateInfoValue    ::= CAKeyUpdAnnContent
--   id-it-currentCRL       OBJECT IDENTIFIER ::= {id-it 6}
--      CurrentCRLValue         ::= CertificateList
--   id-it-unsupportedOIDs  OBJECT IDENTIFIER ::= {id-it 7}
--      UnsupportedOIDsValue    ::= SEQUENCE OF OBJECT IDENTIFIER
--   id-it-keyPairParamReq  OBJECT IDENTIFIER ::= {id-it 10}
--      KeyPairParamReqValue    ::= OBJECT IDENTIFIER
--   id-it-keyPairParamRep  OBJECT IDENTIFIER ::= {id-it 11}
```

```
--      KeyPairParamRepValue    ::= AlgorithmIdentifer
--   id-it-revPassphrase    OBJECT IDENTIFIER ::= {id-it 12}
--      RevPassphraseValue      ::= EncryptedValue
--   id-it-implicitConfirm  OBJECT IDENTIFIER ::= {id-it 13}
--      ImplicitConfirmValue    ::= NULL
--   id-it-confirmWaitTime  OBJECT IDENTIFIER ::= {id-it 14}
--      ConfirmWaitTimeValue    ::= GeneralizedTime
--   id-it-origPKIMessage   OBJECT IDENTIFIER ::= {id-it 15}
--      OrigPKIMessageValue     ::= PKIMessages
--   id-it-suppLangTags     OBJECT IDENTIFIER ::= {id-it 16}
--      SuppLangTagsValue       ::= SEQUENCE OF UTF8String
--
-- где
--   id-pkix OBJECT IDENTIFIER ::= {
--      iso(1) identified-organization(3)
--      dod(6) internet(1) security(5) mechanisms(5) pkix(7)}
-- и
--   id-it   OBJECT IDENTIFIER ::= {id-pkix 4}
--
-- Эта конструкция МОЖЕТ использоваться также для определения новых запросов и откликов
-- PKIX CMP или сообщения общего назначения (например, анонсов) в соответствии с
-- будущими потребностями конкретных сред.

GenMsgContent ::= SEQUENCE OF InfoTypeAndValue
-- Может передаваться элементом EE, RA или CA (в зависимости от содержимого).
-- ОПЦИОНАЛЬНЫЙ параметр infoValue в InfoTypeAndValue обычно был опущен в приведённых
-- выше примерах. Получатель волен игнорировать все включённые идентификаторы объектов,
-- которые он не распознал. При передаче от EE к CA пустой набор показывает, что CA
-- может передать любую (всю) информацию, которую он захочет.

GenRepContent ::= SEQUENCE OF InfoTypeAndValue
-- Получатель МОЖЕТ игнорировать все нераспознанные значения OID.

ErrorMsgContent ::= SEQUENCE {
    pKIStatusInfo          PKIStatusInfo,
    errorCode              INTEGER           OPTIONAL,
    -- определяемые реализацией коды ошибок
    errorDetails           PKIFreeText       OPTIONAL
    -- определяемая реализацией информация об ошибке
}

PollReqContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER
}

PollRepContent ::= SEQUENCE OF SEQUENCE {
    certReqId              INTEGER,
    checkAfter             INTEGER,  -- время в секундах
    reason                 PKIFreeText OPTIONAL
}

END — завершение модуля CMP
```

Приложение G. Благодарности

Авторы с благодарностью отмечают вклад членов рабочей группы IETF PKIX и почтовой конференции ICSA CA-talk (созданной для обсуждения вопросов взаимодействия CMP). Усилия этих людей позволили уточнить и сделать более удобной данную спецификацию. Tomi Kause благодарит Vesa Suontama и Toni Tammisalo за их обзор и комментарии.

Адреса авторов

Carlisle Adams

University of Ottawa

800 King Edward Avenue

P.O.Box 450, Station A

Ottawa, Ontario K1N 6N5

CA

Phone: (613) 562-5800 ext. 2345

Fax: (613) 562-5664

EMail: cadams@site.uottawa.ca

Stephen Farrell

Trinity College Dublin

Distributed Systems Group

Computer Science Department

Dublin

IE

Phone: +353-1-608-2945

EMail: stephen.farrell@cs.tcd.ie

Tomi Kause

SSH Communications Security Corp

Valimotie 17

Helsinki 00380

FI

Phone: +358 20 500 7415

EMail: toka@ssh.com

Tero Mononen

SafeNet, Inc.

Fredrikinkatu 47

Helsinki 00100

FI

Phone: +358 20 500 7814

EMail: tmononen@safenet-inc.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Полное заявление авторских прав

Интеллектуальная собственность

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.