Network Working Group
Request for Comments: 4211
Obsoletes: 2511
Category: Standards Track

J. Schaad
Soaring Hawk Consulting

September 2005

Формат сообщений запроса сертификата (CRMF) для инфраструктуры открытых ключей Internet X.509

Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)

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 описаны синтаксис и семантика CRMF[1]. Этот синтаксис служит для передачи запросов сертификата в удостоверяющий центр (CA[2]), возможно через центр регистрации (RA[3]), с целью создания сертификата X.509. Запрос обычно включает открытый ключ и связанные с ним регистрационные данные. Документ не определяет протокол запроса сертификата.

Оглавление

[1]Certificate Request Message Format - формат сообщения запроса сертификата.
[2]Certification Authority - удостоверяющий центр.
[3]Registration Authority - регистрационное агентство, регистратор.

1. Введение и терминология

Этот документ описывает формат сообщений CRMF. Объекты CRM[1] используются в протоколах для передачи запросов удостоверяющим центрам (CA) и, возможно, регистрационным агентствам (RA) на создание сертификатов X.509. Запрос обычно включает открытый ключ и связанную с ним регистрационную информацию.

Определённый в этом документе объект запроса сертификата не является автономным протоколом. Определённая в документе информация предназначена для использования протоколом запроса сертификатов (CRP[2]). Предполагается, что этот протокол определяет используемые алгоритмы, а также набор регистрационных данных и управляющих структур. Большинство требований данного документа относятся к упомянутому протоколу CRP.

Запросы сертификатов могут подаваться агентствами RA, запрашивающими сертификаты от имени субъектов (Subject), удостоверяющими центрами CA, запрашивающими кросс-сертификаты у других CA, или непосредственно конечными объектами (EE[3]).

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 RFC 2119 [RFC2119].

2. Обзор

Создание запроса сертификации включает перечисленные ниже этапы.

a) Создаётся объект CertRequest, который может включать открытый ключ, часть или все имя субъекта, другие запрашиваемые поля сертификата и иную управляющую информацию, относящуюся к процессу регистрации. В зависимости от протокола CRP, эта информация может быть задана субъектом (Subject) и потенциально изменена агентством RA, а также может быть задана агентством RA на основе Subject или представленной субъектом документации.

b) При необходимости рассчитывается доказательство обладания[4] (секретный ключ, соответствующий открытому ключу, для которого запрашивается сертификат).

c) Дополнительная регистрационная информация может объединяться со значением proof-of-possession и структурой CertRequest для формирования CertReqMessage. Дополнительная регистрационная информация может быть добавлена как субъектом, так и регистратором RA.

d) Сообщение CertReqMessage передаётся с использованием защиты в удостоверяющий центр CA. Конкретные меры транспортной защиты определяются каждым протоколом CRP, соответствующим этому документу.

2.1. Отличия от RFC 2511

1. Добавлен вводный раздел.
2. Добавлена концепция CRP и языка, относящегося к протоколам CRP.
3. В параграфе 6.2 обозначение regToken заменено на authenticator.
4. Добавлено описание содержимого структуры EncryptedValue.
5. Изменено имя и содержимое OID {id-regInfo 1}.
6. Добавлена детализация содержимого различных структур, определённых в документе.
7. Заменено Приложение A со ссылкой на [RFC2875]. Единственное отличие состоит в том, прежний текст указывал использование дополнительного имени (alt name) субъекта при пустом основном имени. Такое поведение невозможно для сертификатов CA, введённых с использованием PKIX. Однако будет полезно обновить RFC 2875 с учётом этого обстоятельства.
8. Добавлено Приложение C, описывающее необходимость POP и возможные атаки на POP.[5]
9. Поле pop в структуре CertReqMsg переименовано в popo для предотвращения путаницы между POP и pop.
10. Использование структуры EncryptedValue отменено в пользу структуры EnvelopedData.
11. Более подробно описано структурирование секретных ключей при их шифровании.
12. Для POP в алгоритмах согласования ключей разрешено использование алгоритмов, отличных от DH.

3. Синтаксис CertReqMessage

Сообщение запроса сертификата состоит из запроса сертификата, необязательного поля доказательства обладания (proof-of-possession) и необязательного поля регистрационной информации.

```
CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
   certReq   CertRequest,
   popo      ProofOfPossession  OPTIONAL,
   -- содержимое зависит от типа ключа
   regInfo   SEQUENCE SIZE(1..MAX) of AttributeTypeAndValue OPTIONAL
}
```

Смысл полей CertReqMsg описан ниже.

[1] Certificate Request Message - сообщение запроса сертификата.
[2] Certificate Request Protocol - протокол запроса сертификата.
[3] End Entity - конечный объект.
[4] Proof-of-possession (POP) - доказательство обладания.
[5] В оригинале нумерация после п. 7 ошибочна. См. https://www.rfc-editor.org/errata_search.php?eid=2595. *Прим. перев.*

certReq содержит шаблон запрашиваемого сертификата. Шаблон заполняется субъектом или от его имени. Не все поля шаблона обязательны для заполнения, более подробное описание приведено в разделе 5.

popo содержит значение, используемое для демонстрации того, что лицо (сущность), указанное для сертификата в качестве Subject, реально обладает соответствующим секретным ключом. Это поле меняется по структуре и содержимому в зависимости от алгоритма с открытым ключом и режима его использования (шифрование или подпись), как указывается в поле эмитируемого сертификата KeyUsage. Дополнительная информация приведена в разделе 4.

В поле **regInfo следует** включать только дополнительную информацию, относящуюся к контексту запроса сертификата, когда такая информация нужна для обработки запроса. Эта информация может включать контактные данные подписчика, платёжную (учёт услуг) информацию или иные сведения, полезные для обработки запроса.

Информацию, относящуюся непосредственно к содержимому сертификата, **следует** включать в содержимое certReq. Однако включение в certReq дополнительного содержимого агентствами RA может привести к утрате корректности поля popo (в зависимости от используемого метода POP). Следовательно, данные, предназначенные для содержимого сертификата, **можно** включать в поле regInfo.

Упомянутые выше протоколы CRP отвечают за определение деталей, которые могут быть заданы в поле regInfo. Этот документ описывает один способ представления информации в этом поле, подробно описанный в Приложении A.

4. Доказательство обладания (POP)

Для того, чтобы предотвратить возможность некоторых атак (см. Приложение C) и позволить CA/RA проверять приемлемость привязки между субъектом и парой ключей, заданные здесь структуры управления PKI предоставляют субъекту возможность подтвердить своё владение (и возможность использования) секретным ключом, соответствующим открытому ключу, для которого запрашивается сертификат. Данный CRP свободен в выборе способа реализации POP (например, процедурные меры за пределами основного канала против сообщений CRMF в основной полосе) в своих сертификационных обменах. В данном CRP, агентства CA и RA свободны в выборе между обеспечиваемыми методами POP (т. е., это вопрос локальной политики RA/CA). CRP **следует** определить требуемые методы POP или указать механизм, с помощью которого клиенты могут определить поддерживаемые методы POP.

Любой протокол CRP, соответствующий данному документу, **должен** обеспечивать исполнение POP тем или иным способом. В настоящее время применяется множество протоколов без PKIX (например, различные протоколы электронной почты), которые явно не проверяют привязку между субъектом и секретным ключом. Пока нет работающих протоколов, которые проверяют такую привязку (для ключевых пар подписи, шифрования и согласования ключей) и существуют неоднозначности, нельзя полагаться на то, что привязка подтверждена CA/RA. Следовательно, нельзя достоверно узнать, является ли привязка открытого ключа в сертификате действительно корректной.

POP обеспечивается разными способами в зависимости от типа ключа, для которого запрошен сертификат. Если ключ может применяться с разными целями (например, ключ RSA для подписи и расшифровки), **можно** использовать любой из методов. Разработчики протоколов должны принимать во внимание возможность наличия аппаратных ограничений (например, секретный ключ может храниться в аппаратном токене).

Данная спецификация разрешает случаи, когда POP подтверждается агентством CA, RA или обоими. Некоторые политики требуют от CA проверки POP на момент эмиссии сертификата, а таких случаях агентство RA **должно** пересылать поля CertRequest и ProofOfPossession от конечного элемента в CA без изменений (в такой ситуации RA может проверить POP и отвергнуть запрос сертификата вместо его пересылки в CA). Если использование CA не требуется политикой для проверки POP, агентству RA **следует** переслать запрос конечного элемента и представленные подтверждения без каких-либо изменений в CA, как указано выше. Если это невозможно (например, если RA проверяет POP с использованием специальных каналов), RA использует элемент raVerified для подтверждения агентству CA факта проверки требуемого подтверждения. Если CA/RA использует отдельный канал (out-of-band) для проверки POP (например, физическое представление созданных CA/RA секретных ключей), поле ProofOfPossession опускается.

```
ProofOfPossession ::= CHOICE {
    raVerified        [0] NULL,
    signature         [1] POPOSigningKey,
    keyEncipherment   [2] POPOPrivKey,
    keyAgreement      [3] POPOPrivKey }
```

Поля ProofOfPossession описаны ниже.

raVerified показывает, что агентство RA выполнило процедуру POP, требуемую для запроса сертификата. Это поле используется, если 1) CA не требует подключения себя к проверке POP и 2) RA требуется изменить содержимое поля certReq. Протоколы CRP **должны** обеспечивать для RA метод подписывания ProofOfPossession. Запрашивающему **недопустимо** устанавливать это поле, а RA/CA **недопустимо** воспринимать ProofOfPossession, если запрашивающий установил это поле.

signature служит для выполнения процедуры POP с ключами подписи. Более подробное описание приведено в параграфе 4.1.

keyEncipherment служит для выполнения процедуры POP с ключами шифрования ключей (например, RSA). Более подробное описание приведено в параграфе 4.2.

keyAgreement служит для выполнения процедуры POP с ключами согласования ключей (например, DH). Более подробное описание приведено в параграфе 4.3.

4.1. POP для ключа подписи

POP для ключа подписи выполняется путём подписывания части данных, содержащих отождествление (identity), для которого нужен сертификат.

При выполнении процедуры POP для ключа подписи следует рассмотреть три указанных ниже случая.

1. Субъект сертификата ещё не организовал аутентифицированного отождествления в CA/RA, но имеет пароль и строку отождествления от CA/RA. В этом случае структура POPOSigningKeyInput будет заполняться с использованием publicKeyMAC для authInfo, а пароль и отождествление будут использоваться для расчёта publicKeyMAC. Открытый ключ для запрашиваемого сертификата будет помещён в обе структуры POPOSigningKeyInput и Certificate Template. Поле подписи рассчитывается для DER-представления структуры POPOSigningKeyInput.

2. CA/RA имеет аутентифицированное отождествление для субъекта сертификата, но запрашивающая сертификат сторона не указала это отождествление в запросе. В этом случае структура POPOSigningKeyInput заполняется с использованием sender для authInfo. Открытый ключ для запрашиваемого сертификата будет помещён в обе структуры POPOSigningKeyInput и Certificate Template. Поле подписи рассчитывается для DER-представления структуры POPOSigningKeyInput.

3. Субъект сертификата поместил своё имя в структуру Certificate Template вместе с открытым ключом. В этом случае поле poposkInput структуры POPOSigningKey опускается. Поле подписи рассчитывается для DER-представления certReq[1].

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

Поля POPOSigningKey описаны ниже.

poposkInput содержит данные для подписи, когда они есть. Это поле **должно** присутствовать, если шаблон сертификата не включает оба значения открытого ключа и имени субъекта.

algorithmIdentifier указывает алгоритм подписи и связанные с ним параметры, используемые для создания значения POP.

signature содержит создаваемое значение POP[2]. При наличии poposkInput подпись рассчитывается для DER-представления poposkInput., а при отсутствии для расчёта используется DER-представления certReq.

```
POPOSigningKeyInput ::= SEQUENCE {
    authInfo            CHOICE {
        sender              [0] GeneralName,
        -- используется только при наличии аутентифицированного отождествления
        -- для отправителя (например, DN из выданного ранее и действительного сертификата)
        publicKeyMAC        PKMACValue },
        -- используется при отсутствии для отправителя аутентифицированного GeneralName;
        -- publicKeyMAC содержит основанное на пароле значение MAC
        -- для DER-представления publicKey
    publicKey           SubjectPublicKeyInfo }  -- из CertTemplate
```

Поля POPOSigningKeyInput описаны ниже.

sender указывает аутентифицированное отождествление, которое заранее было организовано для субъекта.

publicKeyMAC содержит значение, рассчитываемое с помощью общего для CA/RA и запрашивающей стороны ключа.

publicKey содержит копию открытого ключа из шаблона сертификата. Копия **должна** в точности совпадать со значением в шаблоне сертификата.

```
PKMACValue ::= SEQUENCE {
   algId  AlgorithmIdentifier,
   value  BIT STRING }
```

Поля PKMACValue описаны ниже.

algId указывает алгоритм, применяемый для расчёта значения MAC. Все реализации **должны** поддерживать алгоритм id-PasswordBasedMAC, более подробно описанных в параграфе 4.4.

value содержит рассчитанное значение MAC, которое вычисляется для DER-представления открытого ключа субъекта сертификата.

Агентство CA/RA идентифицирует общий секрет для использования путём просмотра 1) поля имени в запросе сертификата или 2) любого из значений regToken (параграф 6.1) и authToken (параграф 6.2).

4.2. POP ключа шифрования ключей

Процедура POP для ключа шифрования ключей выполняется одним из трёх различающихся методов. Можно представить секретный (private) ключ в CA/RA, расшифровать зашифрованный вызов от CA/RA (прямой метод) или создать сертификат, который может быть возвращён в зашифрованном виде и возвращён в качестве отклика на вызов (непрямой метод).

```
POPOPrivKey ::= CHOICE {
    thisMessage       [0] BIT STRING,   -- устарело
    subsequentMessage [1] SubsequentMessage,
    dhMAC             [2] BIT STRING,   -- устарело
    agreeMAC          [3] PKMACValue,
    encryptedKey      [4] EnvelopedData }
  -- для keyAgreement (исключительно) владение подтверждается в этом сообщении
  -- (которое содержит MAC (для DER-представления параметра certReq в
  -- CertReqMsg, который может включать subject и publicKey) на основе ключа,
  -- созданного из секретного ключа DH конечного элемента и открытого ключа DH
  -- агентства CA); значение dhMAC ДОЛЖНО рассчитываться в соответствии с RFC 2875
```

[1] В оригинале ошибочно сказано «шаблона сертификата». См. https://www.rfc-editor.org/errata_search.php?eid=4797. *Прим. перев.*
[2] В оригинале допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=166. *Прим. перев.*

```
    -- для статического подтверждения владения.

SubsequentMessage ::= INTEGER {
    encrCert (0),
    challengeResp (1) }
```

Поля POPOPrivKey описаны ниже.

thisMessage содержит зашифрованный секретный ключ, для которого выдаётся сертификат. Владение секретным ключом подтверждается его предоставлением в CA/RA. При подготовке первой спецификации это поле было описано некорректно. Правильным назначением этого поля является создание структуры EncryptedValue, в которой зашифрованное содержимое является секретным ключом, а сама структура EncryptedValue затем «оборачивается» (wrapped) в тип BIT STRING. От этого поля отказались в пользу encryptedKey.

subsequentMessage используется для индикации выполнения POP путём расшифровки сообщения от CA/RA и возврата результат. Тип сообщения для расшифровки указывается использованным значением (value).

encrCert указывает, что выданный сертификат будет возвращён в зашифрованном виде. Запрашивающий должен будет расшифровать сертификат и предъявить результат агентству CA/RA. Детали обеспечиваются протоколом CRP.

challengeResponse указывает, что от CA/RA запрашивающему было отправлено сообщение с вызовом. Детали этого сообщения и ответа на вызов обеспечиваются протоколом CRP.

dhMAC используется для ключей при согласовании Diffie-Hellman и содержит рассчитанное значение MAC, которое получается с использованием секретного ключа запрашивающего и открытого ключа CA/RA. От применения этого поля отказались в пользу agreeMAC (см. параграф 4.3).

agreeMAC используется для ключей согласования ключа и содержит рассчитанное значение MAC, которое получается с использованием секретного ключа запрашивающего и открытого ключа CA/RA (см. параграф 4.3).

macAlg указывает алгоритм, используемый для расчёта значения MAC.

macValue содержит рассчитанное значение MAC.

encryptedKey содержит зашифрованный секретный ключ, соответствующий открытому ключу, для которого выдаётся сертификат, а также идентификационное значение для указания факта создания стороной, запрашивающей сертификат. Зашифрованное содержимое **должно** иметь тип id-ct-encKeyWithID.

Предполагается, что протоколы, соответствующие этой спецификации, будут включать механизмы подтверждений и вызовов-откликов для завершения работы протокола.

4.2.1. Тип содержимого Private Key Info

Этот тип содержимого служит для 1) подтверждения владения секретными ключами и 2) депонирования секретных ключей (с использованием элемента управления options control, как описано в параграфе 6.4). Эта структура основана на структуре информации о секретном ключе из [PKCS8], но имеет одно преднамеренное отличие. Возможны атаки на агентов депонирования, если те расшифровывают секретные ключи, но не знают, кому принадлежит зашифрованный ключ. Атакующие может перехватить зашифрованный секретный ключ, создать на его основе запрос сертификата и затем запросить операцию восстановления секретного ключа.

Этот тип содержимого и его структура показаны ниже.

```
id-ct-encKeyWithID OBJECT IDENTIFIER ::= {id-ct 21}

EncKeyWithID ::= SEQUENCE {
  privateKey           PrivateKeyInfo,
  identifier CHOICE {
    string               UTF8String,
    generalName          GeneralName
  } OPTIONAL
}

PrivateKeyInfo ::= SEQUENCE {
   version                   INTEGER,
   privateKeyAlgorithm       AlgorithmIdentifier,
   privateKey                OCTET STRING,
   attributes                [0] IMPLICIT Attributes OPTIONAL
}

Attributes ::= SET OF Attribute
```

Ниже описаны поля структуры EncKeyWithID.

privateKey содержит зашифрованный секретный ключ. Данный документ включает определения трёх форматов секретных ключей. Спецификации асимметричных алгоритмов должны для согласованности включать определения как секретных, так и открытых ключей.

identifier содержит имя, которое агентство CA/RA может связать с запрашивающим. Обычно это часть любого имени субъекта, его дополнительное имя (из имеющегося сертификата или запроса сертификата) или текстовый маркер, известный запрашивающему и CA/RA[1]. Это поле **должно** присутствовать, если цель заключается в подтверждении владения секретным ключом. Поле **следует** включать, если предполагается архивирование ключа и его расшифровка агентом архивирования.

Ниже описаны поля структуры PrivatekeyInfo.

version должно иметь значение 0.

[1]В исходном документе здесь допущена ошибка. См. https://www.rfc-editor.org/errata_search.php?eid=2340. *Прим. перев.*

privateKeyAlgorithm содержит идентификатор объекта для секретного ключа.

privateKey строка октетов, содержащая секретный ключ в формате, определяемом значением privateKeyAlgorithm.

attributes представляет собой набор атрибутов, которые содержат расширенную информацию о секретном ключе.

4.2.2. Структуры секретных ключей

Ниже определены структуры для использования с тремя алгоритмами.

4.2.2.1. Ключи D-H

При создании структуры PrivateKeyInfo для ключа D-H применяются приведённые ниже правила.

1. Поле privateKeyAlgorithm **должно** иметь значение id-dh-private-number. Параметр id-dh-private-number представляет собой значение DomainParameters (импорт из [PKIXALG]).
2. Структура ASN для privateKey **должна** иметь форму

```
DH-PrivateKey ::= INTEGER
```

3. Поле attributes **должно** быть опущено.

4.2.2.2. Ключи DSA

При создании структуры PrivateKeyInfo для ключа DSA применяются приведённые ниже правила.

1. Поле privateKeyAlgorithm **должно** иметь значение id-dsa. Параметрами id-dsa являются Dss-Parms (импорт из [PKIXALG]).
2. Структура ASN для privateKey **должна** иметь форму

```
DSA-PrivateKey ::= INTEGER
```

3. Поле attributes **должно** быть опущено.

4.2.2.3. Ключи RSA

При создании структуры PrivateKeyInfo для ключа RSA применяются приведённые ниже правила.

1. Поле privateKeyAlgorithm **должно** иметь значение rsaEncryption.
2. Структура ASN для privateKey MUST **должна** быть RSAPrivateKey (определено в [PKCS1]).
3. Поле attributes **должно** быть опущено.

4.2.3. Рекомендации для вызовов-откликов

Ниже приводятся рекомендации для разработчиков протоколов регистрации в части ожиданий по работе непрямого подтверждения владения (indirect proof-of-possession0 и о некоторых «подводных камнях» при создании сообщений для реализации этого метода POP.

1. Исходный запрос на регистрацию включает то или иное подтверждение идентификации и открытую часть ключа шифрования. Отметим, что подтверждение идентификации должно покрывать открытую часть ключа шифрования для предотвращения атак с подменой ключа, когда атакующий включает свой открытый ключ взамен вашего.
2. Отклик от сервера включает зашифрованное значение того или иного типа данных. Получение этого значения от сервера нужно аутентифицировать каким-либо способом. Спецификация должна включать детальное описание способа возврата этого значения для разных типов ключей. Для ключей RSA значение может быть указано, как шифруемое непосредственно с открытым ключом RSA, но такой подход не будет работать с ключом D-H, где требуется указать опосредованный механизм шифрования значения.
3. Второе сообщение запроса включает хэш расшифрованного значения. Здесь **недопустимо** указывать просто хэш зашифрованного значения, а также никогда не следует «подписывать» полностью случайное значение. Желательно включение в процесс хэширования информации типа строки идентификации, чтобы это можно было сделать явно. Это возвращаемое значение **должно** включаться во второе подтверждение идентификации.

При выполнении непрямой процедуры POP настоятельно рекомендуется требовать идентификаторы транзакций и одноразовые значения nonce, что позволит 1) связать между собой разные сообщения процесса и 2) внести каждому элементу некоторые случайные элементы в процесс подтверждения идентификации.

4.3. POP для ключа согласования ключей

POP для ключа согласования ключей выполняется одним из 4 разных способов. Три первых способа идентичны представленным выше для ключа шифрования ключей. Преимуществом четвёртого метода является создание общего секрета и возможность его использования для информации MAC.

При использовании представленных выше прямых или опосредованных методов шифрования агентству CA/RA потребуется создавать для таких случаев эфемерный ключ, если параметры алгоритма шифрования CA/RA и запрашивающей стороны не совпадают.

Конечный объект может запросить MAC сертификата (используя полученный расчётным путём общий секрет) в качестве четвёртого варианта выполнения POP. Этот вариант может использоваться только в тех случаях, когда CA/RA уже имеет сертификат, который относится к конечному объекту и субъект (Subject) сертификата может использовать параметры CA/RA.

Для алгоритма согласования ключей DH все реализации **должны** поддерживать статический механизм DH Proof-of-Possession. Подробное описание этого алгоритма приведено в разделе 3 of [RFC2875]. Следует подчеркнуть, что при пустом значении субъекта или эмитента сертификата взамен следует использовать его дополнительное имя.

4.4. Использование MAC на базе паролей

Этот алгоритм MAC был разработан для того, чтобы можно было взять общий секрет (пароль) и использовать его для расчёта проверочного значения применительно к блоку информации. Идея метода заключается в том, что без пароля невозможно рассчитать корректное проверочное значение. Алгоритм многократно использует необратимую функцию с целью замедления всех возможных атак с использованием словарей.

Идентификатор алгоритма и структура параметров для Password-Based MAC приведены ниже.

```
id-PasswordBasedMAC OBJECT IDENTIFIER ::= { 1 2 840 113533 7 66 13}

PBMParameter ::= SEQUENCE {
   salt                OCTET STRING,
   owf                 AlgorithmIdentifier,
   iterationCount      INTEGER,
   mac                 AlgorithmIdentifier
   }[1]
```

Назначение полей PBMParameter[2] описано ниже.

salt содержит случайное значение, используемое для расчёта ключа в процессе MAC. **Следует** применять значения salt размером не менее 8 октетов (64 бита).

owf указывает алгоритм и связанные с ним параметры, используемые для расчёта ключа в процессе MAC. Все реализации **должны** поддерживать алгоритм SHA-1.

iterationCount указывает число итераций хэширования в процессе расчёта ключа и **должно** иметь значение не менее 100 (многие предлагают использовать не менее 1000 итераций). Компромисс здесь заключается в выборе между обеспечиваемым уровнем защиты от атак и временем, затрачиваемым сервером на обработку всех итераций при расчёте паролей. Хэширование обычно считается «недорогой» операцией, но это допущение может стать ошибочным с новыми вариантами хэш-функций.

mac указывает алгоритм и связанные с ним параметры функции MAC. Все реализации **должны** поддерживать алгоритм HMAC-SHA1 [HMAC]. Всем реализациям **следует** также поддерживать DES-MAC и Triple-DES-MAC [PKCS11].

Ниже приведён псевдокод алгоритма.

Входные данные:

pw - строка октетов, содержащая пользовательский пароль;

data - строка октетов, содержащая значение, для которого рассчитывается MAC;

Iter - счётчик итераций.

Выход:

MAC - строка октетов, содержащая полученное в результате значение MAC.

1. создаётся случайное значение salt = S;
2. salt добавляется в конец pw - K = pw || salt;
3. вычисляется хэш от K - K = HASH(K);
4. если Iter > 0, выполняется декрементирование (Iter = Iter - 1) и возврат к п. 3;
5. Рассчитывается HMAC в соответствии с [HMAC]

   `MAC = HASH( K XOR opad, HASH( K XOR ipad, data) )`
   значения opad и ipad определены в [HMAC].

5. Синтаксис CertRequest

Синтаксис CertRequest включает идентификатор запроса, шаблон содержимого сертификата и необязательную последовательность управляющей информации.

```
CertRequest ::= SEQUENCE {
   certReqId     INTEGER,             -- ID для сопоставления запросов с откликами
   certTemplate  CertTemplate,        -- выбранные поля выпускаемого сертификата
   controls      Controls OPTIONAL } -- атрибуты, влияющие на выпуск сертификата

CertTemplate ::= SEQUENCE {
   version      [0] Version               OPTIONAL,
   serialNumber [1] INTEGER               OPTIONAL,
   signingAlg   [2] AlgorithmIdentifier   OPTIONAL,
   issuer       [3] Name                  OPTIONAL,
   validity     [4] OptionalValidity      OPTIONAL,
   subject      [5] Name                  OPTIONAL,
   publicKey    [6] SubjectPublicKeyInfo  OPTIONAL,
   issuerUID    [7] UniqueIdentifier      OPTIONAL,
   subjectUID   [8] UniqueIdentifier      OPTIONAL,
```

[1] В оригинале опечатка -). См. https://www.rfc-editor.org/errata_search.php?eid=2341. *Прим. перев.*
[2] В оригинале опечатка - PEMParameter. См. https://www.rfc-editor.org/errata_search.php?eid=2342. *Прим. перев.*

```
      extensions   [9] Extensions            OPTIONAL }

   OptionalValidity ::= SEQUENCE {
      notBefore  [0] Time OPTIONAL,
      notAfter   [1] Time OPTIONAL } -- требуется наличие хотя бы одного

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

Поля CertRequest описаны ниже.

certReqId содержит целочисленное значение, используемое запросившей сертификат стороной для сопоставления запроса с откликом.

certTemplate содержит шаблон сертификата X.509. Запрашивающий указывает в этих полях конкретные желаемые значения. Более подробное описание полей приведено ниже.

controls содержит атрибуты, которые не являются частью сертификата, но контролируют контекст, в котором выпускается сертификат. Подробное описание элементов управления, определённых в этом документе, можно найти в разделе 6. В других документах могут определяться свои элементы управления. За спецификацию требуемых элементов управления отвечает протокол CRP.

Поля CertTemplate описаны ниже.

version должно иметь значение 2, если версия указана; **следует** опускать это поле.

serialNumber должно опускаться; это значение выделяется агентством CA при создании сертификата.

signingAlg должно опускаться; это значение выделяется агентством CA при создании сертификата.

issuer обычно опускается; поле заполняется агентством CA, в котором запрашивающий желает выпустить сертификат, в ситуации когда RA обслуживается несколькими CA.

validity обычно опускается, но может применяться для запроса конкретных будущих дат начала и завершения срока действия сертификата. Наиболее часто это поле будет использоваться при кросс-сертификации для CA. В таких случаях в этом поле может выть указан срок действия имеющегося сертификата и тогда новый сертификат будет иметь такой же срок действия. Если поле validity присутствует, в нем **должно** быть задано хотя бы одно из двух следующих субполей:

notBefore указывает время начала действия сертификата, заданное по тем же правилам, которые применяются для notBefore в [PROFILE];

notAfter указывает время завершения действия сертификата, заданное по тем же правилам, которые применяются для notAfter в [PROFILE].

subject содержит предложенное имя для запрашивающего, в качестве которого обычно указывается то имя, которое ранее было присвоено запрашивающему агентством CA.

publicKey содержит открытый ключ, для которого создаётся сертификат. Это поле **должно** быть заполнено, если запрашивающим сам генерирует ключ, и опускается в случаях генерации ключа агентством RA/CA.

issuerUID должно быть опущено; это поле отменено в [PROFILE].

subjectUID должно быть опущено; это поле отменено в [PROFILE].

extensions указывает расширения, которые запрашивающий хочет поместить в сертификат. Обычно эти расширения связаны с такими вещами, как установка для ключа использования в качестве keyEncipherment.

Агентству CA/RA разрешено менять значения любого поля, за исключением publicKey. Возвращённый сертификат должен быть проверен запрашивающим на предмет приемлемости установленных значений полей. Агентствам CA/RA **следует** по возможности применять шаблоны.

В некоторых случаях все поля шаблона могут быть опущены. Если генерация ключа происходила в CA/RA и отождествление было помещено в другое место (например, как в описанном ниже id-regCtrl-regToken), запрашивающему не требуется задавать какое-либо поля.

6. Синтаксис элементов управления

При генерации CertRequest в запрос может быть включено одно или несколько управляющих значений, относящихся к обработке запроса.

```
   Controls  ::= SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue
```

В этом документе определено несколько элементов управления - regToken (параграф 6.1), authenticator (параграф 6.2), pkiPublicationInfo (параграф 6.3), pkiArchiveOptions (параграф 6.4), oldCertID (параграф 6.5), protocolEncrKey (параграф 6.6). Каждый протокол CRP **должен** определять набор поддерживаемых им элементов управления. Дополнительные элементы могут определяться в других RFC или в самом протоколе CRP.

6.1. Элемент regToken

Элемент управления regToken содержит одноразовую информацию (создаётся на основе секретного значения или иной общей информации), предназначенную для использования агентством CA с целью проверки отождествления субъекта до выпуска сертификата. При получении сертификационного запроса с regToken агентство CA проверяет информацию для подтверждения заявленного в запросе отождествления.

Значение regToken может быть создано в CA и передано подписчику по отдельному каналу или может быть создано иным способом, доступным для подписчика и CA. Безопасность любого обмена по отдельному каналу должна соизмеряться с риском восприятия агентством CA перехваченного значения от кого-то другого вместо предполагаемого

подписчика. Значение regToken не шифруется при возврате и, если требуется конфиденциальность, она должна быть обеспечена запрашивающей стороной.

Элемент regToken используется только для инициализации конечного элемента в PKI, тогда как элемент authenticator (см. параграф 6.2[1]) может применяться как для начального, так и для последующих запросов.

В некоторых вариантах применения элемент regToken может содержать символьную строку или численное значения (типа случайной величины). В последнем случае значение представляется в форме строки символов, соответствующей двоичному значению. В качестве кодировки для regToken **нужно** применять UTF8String.

```
id-regCtrl-regToken OBJECT IDENTIFIER ::= { id-regCtrl 1 }
```

Без предварительного согласования между подписчиком и CA это значение будет содержать текстовое представление некого общего секрета. Если вместо этого используется значения, рассчитанное на основе общего секрета, предполагается, что протокол CRP определяет новый элемент управления для этого конкретного расчёта.

6.2. Элемент authenticator

Элемент управления authenticator содержит информацию, используемую на постоянной основе для организации некриптографической проверки отождествления при коммуникациях с CA. Примерами могут служить девичья фамилия матери, последние четыре цифры номера социального страхования или другая информация, доступная подписчику и CA, хэш-значение такой информации или иные данные, созданные специально для этой цели. Значение для элемента authenticator может создаваться подписчиком или агентством CA.

В некоторых вариантах применения элемент authenticator может содержать символьную строку или численное значения (типа случайной величины). В последнем случае значение представляется в форме строки символов, соответствующей двоичному значению. В качестве кодировки для authenticator **нужно** применять UTF8String.

```
id-regCtrl-authenticator OBJECT IDENTIFIER ::= { id-regCtrl 2 }
```

При выборе между использованием элемента authenticator или regToken следует использовать приведённые ниже рекомендации. Если используется однократное значение, следует выбирать элемент regToken. Если значение предназначено для долговременного использования, следует выбирать элемент authenticator.

6.3. Элемент pkiPublicationInfo

Элемент pkiPublicationInfo обеспечивает подписчикам возможность влиять на публикацию сертификата агентством CA/RA. Этот элемент считается рекомендацией и CA/RA могут игнорировать его. Элемент определяется приведённым ниже OID и синтаксисом.

```
id-regCtrl-pkiPublicationInfo  OBJECT IDENTIFIER ::= { id-regCtrl 3 }

PKIPublicationInfo ::= SEQUENCE {
     action     INTEGER {
                  dontPublish (0),
                  pleasePublish (1) },
     pubInfos  SEQUENCE SIZE (1..MAX) OF SinglePubInfo OPTIONAL }

SinglePubInfo ::= SEQUENCE {
      pubMethod    INTEGER {
          dontCare    (0),
          x500        (1),
          web         (2),
          ldap        (3) },
      pubLocation  GeneralName OPTIONAL }
```

Ниже описано назначение полей структуры PKIPublicationInfo.

action показывает желание или нежелание запрашивающей стороны публикации сертификата агентством CA/RA. Возможны значения приведены ниже[2]:

dontPublish говорит о желании запрашивающего отказаться от публикации сертификата (это может говорить о намерении опубликовать сертификат самостоятельно). При использовании значения dontPublish поле pubInfos **должно** быть опущено.

pleasePublish говорит о желании запрашивающего опубликовать сертификат через агентство CA/RA.

pubInfos указывает место, где запрашивающий желает опубликовать сертификат с помощью CA/RA. Это поле опускается при выборе запрашивающим опции dontPublish. Если запрашивающий хочет задать некие места публикации сертификата и позволяет CA/RA публиковать его в других местах, он будет указывать в структуре SinglePubInfo множество значений, одним из которых является dontCare.

Ниже описано назначение полей структуры SinglePubInfo.

pubMethod указывает тип адреса места, куда запрашивающий желает поместить сертификат с участием CA/RA.

dontCare показывает, что CA/RA может публиковать сертификат в любом месте по своему выбору. При указании dontCare поле pubLocation[3] **должно** быть опущено.

x500 показывает желание запрашивающего опубликовать сертификат с помощью CA/RA в указанном месте. Место публикации задаётся полем x500 в структуре pubLocation.

ldap показывает желание запрашивающего опубликовать сертификат с помощью CA/RA в указанном месте. Место публикации задаётся полем ldap в структуре pubLocation.

[1] В оригинале ошибочно сказано 7.2. См. https://www.rfc-editor.org/errata_search.php?eid=2343. *Прим. перев.*
[2] В оригинале опечатка. См. https://www.rfc-editor.org/errata_search.php?eid=2344. *Прим. перев.*
[3] В оригинале опечатка - pubInfos. См. https://www.rfc-editor.org/errata_search.php?eid=2345. *Прим. перев.*

web показывает желание запрашивающего опубликовать сертификат с помощью CA/RA в указанном месте. Место публикации задаётся полем http в структуре pubLocation.

pubLocation содержит адрес, по которому будет публиковаться сертификат. Выбор значения поля GeneralName определяется значением pubMethod в этой структуре.

Места для публикации могут указываться в любом порядке. Все указанные места обрабатываются CA в целях публикации.

6.4. Элемент pkiArchiveOptions

Элемент pkiArchiveOptions обеспечивает подписчикам возможность представить информацию, требуемую для организации архива секретных ключей, соответствующих открытым ключам в запросах сертификатов. Идентификатор OID и синтаксис элемента представлены ниже.

```
id-regCtrl-pkiArchiveOptions   OBJECT IDENTIFIER ::= { id-regCtrl 4 }

PKIArchiveOptions ::= CHOICE {
   encryptedPrivKey     [0] EncryptedKey,
   -- актуальное значение секретного ключа
   keyGenParameters     [1] KeyGenParameters,
   -- параметры, позволяющие заново сгенерировать секретный ключ
   archiveRemGenPrivKey [2] BOOLEAN }
   -- устанавливается значение TRUE, если отправитель хочет, чтобы получатель
   -- архивировал секретный ключ пары, которую он генерирует в ответ на данный запрос;
   -- FALSE, если архивирование нежелательно.

EncryptedKey ::= CHOICE {
   encryptedValue        EncryptedValue, -- устарело
   envelopedData     [0] EnvelopedData }
   -- зашифрованный секретный ключ ДОЛЖЕН помещаться в строку октетов
   -- envelopedData encryptedContentInfo encryptedContent.

EncryptedValue ::= SEQUENCE {
   intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
   -- предусмотренный алгоритм, с которым будет использоваться значение
   symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
   -- симметричный алгоритм шифрования значения
   encSymmKey    [2] BIT STRING           OPTIONAL,
   -- (зашифрованный) симметричный ключ шифрования значения
   keyAlg        [3] AlgorithmIdentifier  OPTIONAL,
   -- алгоритм для шифрования симметричного ключа
   valueHint     [4] OCTET STRING         OPTIONAL,
   -- краткое описание или идентификатор содержимого encValue
   -- (может быть осмысленным только для передающей стороны и применяться лишь
   -- в тех случаях, когда EncryptedValue можно перепроверить в будущем
   -- на передающей стороне)
   encValue       BIT STRING }
-- От применения поля EncryptedValue отказались в пользу структуры EnvelopedData.
--
-- При использовании EncryptedValue для передачи секретного ключа (в отличие от
-- сертификата), реализация ДОЛЖНА поддерживать поле encValue, содержащее
-- зашифрованное значение PrivateKeyInfo как указано в параграфе 12.11 [PKCS11].
-- Если encValue содержит иной формат/кодирование секретного ключа, первый октет
-- valueHint МОЖЕТ применяться для индикации этого формата/кодирования (отметим,
-- что возможные значения этого октета в настоящее время не заданы). В любом случае
-- поле intendedAlg ДОЛЖНО использоваться для индикации хотя бы идентификатора OID
-- предусмотренного алгоритма секретного ключа, если он не известен заранее отправителю
-- и получателю.

KeyGenParameters ::= OCTET STRING
```

Ниже описано назначение полей структуры PKIArchiveOptions.

encryptedPrivKey содержит зашифрованную версию секретного ключа.

keyGenParameters содержит информацию, требуемую запрашивающему для восстановления секретного ключа. Например, для многих реализаций RSA можно передать первое случайное число, являющееся простым (с проверкой). Указанная здесь структура не определяется данным документом. Протоколы CRP, определяющие содержимое этой структуры, **должны** определять не только структуру, но и способ защиты данных от несанкционированного доступа.

archiveRemGenPrivKey показывает желание запрашивающего архивировать ключ, создаваемый агентством CA/RA от имени запрашивающего.

Ниже описано назначение полей структуры EncryptedKey.

encryptedValue больше не применяется[1]. Это поле было отменено вместе со структурой EncryptedValue.

envelopedData содержит зашифрованное значение секретного ключа. Использующие эту структуру протоколы CPR **должны** определять элемент(ы), для которого данные шифруются (EE, агенты депонирования, УЦ), и способ определения ключа или набора ключей. Подробное описание структуры EnvelopedData приведено в [CMS]. Зашифрованное содержимое **должно** представлять собой id-ct-encKeyWithID. Идентификатор может быть опущен, если эта структура не используется также для доказательства владения.

[1] В оригинале опечатка. См. https://www.rfc-editor.org/errata_search.php?eid=2346. *Прим. перев.*

6.5. Элемент OldCertID

При наличии элемента OldCertID он указывает сертификат, обновляемый текущим запросом. Идентификатор OID и синтаксис приведены ниже.

```
id-regCtrl-oldCertID           OBJECT IDENTIFIER ::= { id-regCtrl 5 }

CertId ::= SEQUENCE {
      issuer           GeneralName,
      serialNumber     INTEGER
  }
```

6.6. Элемент protocolEncrKey

При наличии элемента protocolEncrKey он указывает ключ, который CA использует для шифрования откликов на сообщения CertReqMessage. В качестве OID для этого элемента используется id-regCtrl-protocolEncrKey. В качестве структуры параметра для этого поля применяется SubjectPublicKeyInfo (определена в [PROFILE]).

```
id-regCtrl-protocolEncrKey     OBJECT IDENTIFIER ::= { id-regCtrl 6 }
```

Этот элемент используется в тех случаях, когда у CA имеется информация для передачи подписчику, которую нужно зашифровать. К такой информации относятся секретные ключи, генерируемые CA для использования подписчиками.

7. Элементы RegInfo

В этом разделе описаны элементы управления, которые помещаются в поле regInfo структуры CertReqMsg.

7.1. utf8Pairs

Этот элемент управления служит для передачи текстовой информации из Subject агентству RA или CA, выпускающему сертификат. Для этой структуры используется OID id-regInfo-utf8Paris и тип UTF8String.

```
id-regInfo-utf8Pairs    OBJECT IDENTIFIER ::= { id-regInfo 1 }
```

Имя завершается знаком вопроса (?), а значение символом процента (%). Пары «имя-значение» могут повторяться. Таким образом, синтаксис имеет вид

```
Name?Value%[Name?Value%]*
```

Механизм %xx, описанный в разделе 2 STD 66 [RFC3986][1] позволяет представлять символы ? (%3f) и % (%25) при их использовании не в качестве разделителей. **Недопустимо** использовать имена, начинающиеся с цифры.

Этот элемент может неоднократно включаться в структуру regInfo. При возникновении информационных конфликтов они должны устраняться в соответствии с локальной политикой RA/CA.

В Приложении A приведён список общепринятых имён и форматы данных, соответствующие наиболее часто встречающимся в сертификатах и каталогах полям.

7.2. certReq

Этот элемент управления предназначен для решения проблемы, возникающей в случаях, когда RA требуется изменить шаблон сертификата, предложенный Subject, но этот шаблон используется в Subject при расчёте POP. В таких случаях RA может поместить новый шаблон сертификата в regInfo.

Этот элемент имеет OID id-regInfo-certReq и структуру CertRequest. В regInfo может помещаться только один экземпляр данного атрибута. При наличии этого элемента управления в структуре regInfo шаблон сертификата из запроса игнорируется. Агентство RA **должно** скопировать все данные из основного шаблона в этот атрибут.

```
id-regInfo-certReq       OBJECT IDENTIFIER ::= { id-regInfo 2 }
```

8. Идентификаторы объектов

OID id-pkix имеет значение

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

-- arc для протоколов Internet X.509 PKI и их компонент
id-pkip  OBJECT IDENTIFIER :: { id-pkix pkip(5) }

-- arc для элементов управления Registration в CRMF
id-regCtrl  OBJECT IDENTIFIER ::= { id-pkip regCtrl(1) }

-- arc для Registration Info в CRMF
id-regInfo       OBJECT IDENTIFIER ::= { id-pkip id-regInfo(2) }
```

9. Вопросы безопасности

Протоколы регистрации по своей природе включают большой объем персональных (приватных) данных. Это могут быть ключи, идентификационные номера, номера кредитных карт и т. п. Безопасность любого протокола CRP основана на механизмах защиты протокола и/или процессов, используемый при коммуникациях между CA, RA и EE. Все протоколы должны обеспечивать маскировку (например, с помощью шифрования или отдельной (off-line) обработки) всей «деликатной» пользовательской информации.

Многие протоколы поддерживают проверку первоначального отождествления между CA/RA и EE с помощью маркеров (token). Обычно такой маркер доставляется с использованием отдельных каналов и средств (out-of-band) типа государственной почтовой системы. Защита каналов доставки в этом случае должна соразмеряться с риском перехвата маркера посторонними лицами.

[1] В оригинале ошибочно указан RFC 1738. См. https://www.rfc-editor.org/errata_search.php?eid=2347. *Прим. перев.*

Реализации должны поддерживать значения POP в процессе регистрации сертификатов. Нужен хороший алгоритм POP для доказательства того, что 1) ключ связан с конкретным пользователем и 2) пользователь применяет ключ, о котором идёт речь. Отказ от реализации POP позволяет создавать сертификаты без корректной привязки между открытым ключом и именем. Это делает возможной подмену владельца ключа и перехват шифрованных сообщений.

Реализации должны применять генератор случайных чисел с высоким уровнем энтропии при создании секретных ключей. Реализации должны генерировать случайные ключи шифрования содержимого, ключи аутентификации сообщений, векторы инициализации (IV), «затравки» (salt) и заполнение. Применение низкокачественных генераторов псевдослучайных чисел (PRNG[1]) для создания криптографических ключей может привести к недостаточной защите или её отсутствию. Для атакующего может оказаться проще воспроизвести среду PRNG, использованную для создания ключей, и провести поиск среди сравнительно небольшого числа вариантов, нежели пытаться подобрать нужное значение из всего пространства ключей. Создание качественных случайных чисел является трудной задачей. В RFC 4086 [RANDOM] приведены важные рекомендации по этим вопросам, а Приложение 3 в FIPS Pub 186 [DSS] описывает один из качественных методов PRNG.

Реализации должны защищать секретные ключи. Утечка секретного ключа подписи позволяет другим замаскироваться под владельца подписи. Утечка секретного ключа расшифровки позволяет перехватывать сообщения.

Одной из особенностью синтаксиса запросов сертификата является то, что генерация ключей происходит удалённо по отношению к месту создания запроса сертификата. Это свойство никогда не следует применять для запросов генерации ключей подписи. Если ключи подписи создаются для пользователя, вступает в силу элемент отказа (repudiation). Пользователь может заявить, что объект был подписан создавшей ключ стороной или любой другой стороной, которая могла видеть ключ в процессе его передачи от места создания до EE[2]. Следует принимать меры по защите ключей шифрования удаленным генератором ключей для предотвращения возможности перехвата ключа посторонними. Это означает, что применяемые алгоритмы шифрования должны быть защищены, и должен применяться ключ шифрования содержимого или других ключей для маскировки секретного ключа в процессе его возврата пользователю. Протоколы CRP никогда не должны предполагать, что ключ подписи, созданный пользователем, может применяться для расшифровки пакета, в котором передаётся секретный ключ шифрования.

В этом документе описан метод, с помощью которого может выполняться депонирование ключей. При этом требуется принимать во внимание несколько важных вопросов. Во-первых, клиент должен быть способен корректно указать объект, где будет депонироваться ключ, или протокол CRP должен предоставить метод, с помощью которого клиент может найти эту информацию. Протокол CRP не может предполагать, что агент депонирования ключей совпадает с агентством CA и, следовательно, имеет то же имя. Во-вторых, алгоритмы, применяемые для маскировки секретных ключей или другой, связанной с генерацией ключей, информации в процессе доставки к агенту условного депонирования, должны соответствовать важности (ценности) данных, защищаемых этим ключом. В-третьих, агенты условного депонирования должны быть достаточно защищены, чтобы депонированные ключи возвращались лишь тем, кто вправе и может получать секретный ключ (обычно это лишь тот, кто предал ключ на депонирование). В-четвёртых, база депонированных данных должна храниться защищённым способом. Одним из распространённых вариантов защиты является шифрование данных, обеспечивающее доступ к ним лишь агенту депонирования. В этом случае может потребоваться депонирование ключа самого агента депонирования. Доступ к агенту депонирования или его архивированному ключу будет означать доступ ко всем ключам, депонированным данным агентом.

10. Литература

10.1. Нормативные документы

[PKCS1] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

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

[PKCS11] RSA Laboratories, The Public-Key Cryptography Standards - "PKCS #11 v2.11: Cryptographic Token Interface Standard", RSA Security Inc., June 2001.

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

[PROFILE] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[PKIXALG] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[RFC2875] Prafullchandra, H. and J. Schaad, "Diffie-Hellman Proof-of-Possession Algorithms", RFC 2875, July 2000.

10.2. Дополнительная литература

[DSS] National Institute of Standards and Technology, FIPS Pub 186: Digital Signature Standard, May 1994.

[PKCS8] RSA Laboratories, "PKCS #8: Private-Key Information Syntax Standard", PKCS #8 v1.2, November 1993.

[RANDOM] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

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

[RFC3986][3] T. Berners-Lee, R. Fielding, L. Masinter "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[1] Pseudo-random number generator.
[2] В оригинале опечатка. См. https://www.rfc-editor.org/errata_search.php?eid=2348. *Прим. перев.*
[3] В оригинале ошибочно указан RFC 1738. См. https://www.rfc-editor.org/errata_search.php?eid=2349. *Прим. перев.*

11. Благодарности

Рабочая группа благодарит Michael Myers, Carlisle Adams, Dave Solo и David Kemp, создавших исходный вариант этого документа.

Рабочая группа выражает свою признательность участникам Barbara Fox, Warwick Ford, Russ Housley и John Pawling, чьи рецензии и комментарии сделали эту спецификацию значительно яснее и полезней. Спасибо также участникам почтовой конференции ca-talk за их полезные предложения в части тестирования взаимодействия.

Текст Приложения C (Зачем использовать POP) был взят из почтового сообщения Al Arsenault и изначально являлся частью документа PKIX Roadmap.

Приложение A. Использование RegInfo для пар Name-Value

Поле value строки id-regInfo-utf8Pairs (с полем tag = 12 и подходящим полем length) будет содержать последовательность пар «имя-значение» в формате UTF-8.

В этом приложении приведены некоторые примеры таких пар общего назначения в целях продвижения взаимодействия независимых реализаций данной спецификации. Этот список не является исчерпывающим и будет расти с течением времени и обретением опыта реализации.

A.1. Определённые имена

В таблице приведён рекомендуемый набор именованных элементов. Значения в колонке «Имя» содержат именно те строки, которые будут появляться в regInfo.

Имя	*Описание*
version	Версия данной вариации, которую использует regInfo.
corp_company	Компания подписчика.
org_unit	Подразделение организации.
mail_firstName	Компонента персонального имени.
mail_middleName	Компонента персонального имени.
mail_lastName	Компонента персонального имени.
mail_email	Адрес электронной почты подписчика.
jobTitle	Должность подписчика.
employeeID	Числовой или текстовый идентификатор сотрудника.
mailStop	
issuerName	Название УЦ.
subjectName	Название субъекта.
validity	Срок действия.

Например,

```
version?1%corp_company?Example, Inc.%org_unit?Engineering%
mail_firstName?John%mail_lastName?Smith%jobTitle?Team Leader%
mail_email?john@example.com%
```

A.2. Представление IssuerName, SubjectName и Validity

При включении в id-regInfo-utf8Pairs в качестве именованных элементов для кодирования значений issuerName, subjectName и validity **нужно** использовать описанный ниже синтаксис. Символы [] указывают необязательные поля, ::= и | применяются в обычном для BNF смысле, а все остальные символы (за исключением незначимых пробелов) вне имён являются завершающими (terminal). Регистр символов принимается во внимание.

```
issuerName  ::= <names>
subjectName ::= <names>
<names>     ::= <name> | <names>:<name>

<validity>  ::= validity ? [<notbefore>]-[<notafter>]

<notbefore> ::= <time>
<notafter>  ::= <time>
```

<time> указывает время UTC в формате YYYYMMDD[HH[MM[SS]]]. Компоненты HH, MM и SS по умолчанию имеют значения 00 и опускаются, если за ними нет отличных от 00 символов.

Примером представления validity может служить

```
validity?-19991231%
```

где начало действия (notBefore) не задано, а заканчивается срок действия 31 декабря 1999 года (notAfter).

Каждое имя содержит один символ идентификатора формы, за которым следует значение имени из одного или множества символов UTF-8. Внутри значения имени для устранения неоднозначностей при наличии символов форматирования внешнего уровня **нужно** использовать escape-последовательности вида %xx, где xx указывает шестнадцатеричный код символа. Сам символ процента представляется последовательностью %%.

```
<name> ::= X<xname>|O<oname>|E<ename>|D<dname>|U<uname>|I<iname>
```

Форматы имён описаны ниже.

Форма имени каталога X.500 (идентификатор X)

```
<xname> ::= <rdns>
<rdns>  ::= <rdn> | <rdns> , <rdn>
<rdn>   ::= <avas>
<avas>  ::= <ava> | <avas> + <ava>
<ava>   ::= <attyp> = <avalue>
<attyp> ::= OID.<oid> | <stdat>
```

Стандартный тип атрибута <stdat> представляет собой символьный идентификатор типа атрибута из числа приведённых ниже.

C	(страна)
L	(населённый пункт)
ST	(штат или провинция)
O	(организация)
OU	(подразделение организации)
CN	(имя)
STREET	(адрес)
E	(адрес электронной почты).

Компонента имени <avalue> представляет собой строку UTF-8 размером от 1 до 64 символов с ограничением использования в подмножестве IA5 кодировки UTF - только символов ASN.1 PrintableString.

Другая форма имени (идентификатор O)

```
<oname> ::= <oid> , <utf8string>
```

Адрес электронной почты (rfc822name) (идентификатор E)

```
<ename> ::= <ia5string>
```

Имя DNS (идентификатор D)

```
<dname> ::= <ia5string>
```

Идентификатор URI (идентификатор U)

```
<uname> ::= <ia5string>
```

Адрес IP (идентификатор I)

```
<iname> ::= <oid>
```

Например,

```
issuerName?XOU=Our CA,O=Example,C=US% subjectName?XCN=John Smith,
O=Example, C=US, E=john@example.com%
```

Приложение B. Структуры и идентификаторы ASN.1

```
PKIXCRMF-2005 {iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-crmf2005(36)}

DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
  -- Схема проверки подлинности каталога (X.509)
     Version, AlgorithmIdentifier, Name, Time,
     SubjectPublicKeyInfo, Extensions, UniqueIdentifier, Attribute
        FROM PKIX1Explicit88 {iso(1) identified-organization(3) dod(6)
            internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
            id-pkix1-explicit(18)} -- найдено в [PROFILE]

  -- Расширения сертификата (X.509)
     GeneralName
        FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6)
              internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
              id-pkix1-implicit(19)}  -- найдено в [PROFILE]

  -- Синтаксис криптографических сообщений
     EnvelopedData
        FROM CryptographicMessageSyntax2004 { iso(1) member-body(2)
             us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
             modules(0) cms-2004(24) };  -- найдено в [CMS]

-- Комментарий для приведённого ниже определения можно удалить при использовании
-- компиляторов ASN.1, не понимающих UTF8String.

-- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING
       -- содержимое этого типа соответствует RFC 3629[1].

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

-- arc для протоколов Internet X.509 PKI и их компонент

id-pkip  OBJECT IDENTIFIER ::= { id-pkix 5 }

id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
             us(840) rsadsi(113549) pkcs(1) pkcs9(9) 16 }

id-ct   OBJECT IDENTIFIER ::= { id-smime  1 }  -- типы содержимого
```

[1]В оригинале ошибочно сказано RFC 2279. См. https://www.rfc-editor.org/errata_search.php?eid=2339. *Прим. перев.*

```
-- Основные определения для этого модуля

CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
 certReq   CertRequest,
 popo       ProofOfPossession  OPTIONAL,
 -- содержимое зависит от типа ключа
 regInfo   SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue OPTIONAL }

CertRequest ::= SEQUENCE {
 certReqId     INTEGER,            -- ID для сопоставления запроса и отклика
 certTemplate  CertTemplate,       -- выбранные поля для выпускаемого сертификата
 controls      Controls OPTIONAL } -- атрибуты, влияющие на эмиссию сертификата

CertTemplate ::= SEQUENCE {
 version      [0] Version               OPTIONAL,
 serialNumber [1] INTEGER               OPTIONAL,
 signingAlg   [2] AlgorithmIdentifier   OPTIONAL,
 issuer       [3] Name                  OPTIONAL,
 validity     [4] OptionalValidity      OPTIONAL,
 subject      [5] Name                  OPTIONAL,
 publicKey    [6] SubjectPublicKeyInfo  OPTIONAL,
 issuerUID    [7] UniqueIdentifier      OPTIONAL,
 subjectUID   [8] UniqueIdentifier      OPTIONAL,
 extensions   [9] Extensions            OPTIONAL }

OptionalValidity ::= SEQUENCE {
 notBefore  [0] Time OPTIONAL,
 notAfter   [1] Time OPTIONAL } -- ДОЛЖЕН присутствовать хотя бы один

Controls  ::= SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
 type         OBJECT IDENTIFIER,
 value        ANY DEFINED BY type }
ProofOfPossession ::= CHOICE {
 raVerified        [0] NULL,
 -- используется в тех случаях, когда агентство RA уже убедилось в том, что
 -- запрашивающий владеет секретным ключом
 signature         [1] POPOSigningKey,
 keyEncipherment   [2] POPOPrivKey,
 keyAgreement      [3] POPOPrivKey }

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

 -- Подпись (использует algorithmIdentifier) является DER-представлением
 -- poposkInput. Отметим, что при наличии в CertReqMsg certReq CertTemplate
 -- значений subject и publicKey поле poposkInput ДОЛЖНО опускаться, а
 -- подпись ДОЛЖНА рассчитываться для DER-представления CertReqMsg certReq.
 -- Если CertReqMsg certReq CertTemplate не включает открытого ключа и
 -- подписи (т. е., включает только одно или не содержит обоих), ДОЛЖНО
 -- быть представлено и подписано поле poposkInput.

POPOSigningKeyInput ::= SEQUENCE {
 authInfo            CHOICE {
     sender              [0] GeneralName,
     -- Используется только в тех случаях, когда для отправителя имеется
     -- аутентифицированное отождествление (например, DN из выпущенного
     -- ранее и действительного сейчас сертификата)
     publicKeyMAC        PKMACValue },
     -- Используется в тех случаях, когда для отправителя нет аутентифицированного
     -- GeneralName; publicKeyMAC содержит основанный на пароле код MAC для
     -- DER-представления publicKey
 publicKey           SubjectPublicKeyInfo }  -- из CertTemplate

PKMACValue ::= SEQUENCE {
algId  AlgorithmIdentifier,
-- Значение алгоритма должно быть PasswordBasedMac {1 2 840 113533 7 66 13}
-- Значение параметра - PBMParameter
value  BIT STRING }

PBMParameter ::= SEQUENCE {
   salt                OCTET STRING,
   owf                 AlgorithmIdentifier,
   -- AlgId для необратимой функции (рекомендуется SHA-1)
   iterationCount      INTEGER,
   -- Число итераций применения OWF
   mac                 AlgorithmIdentifier
   -- MAC AlgId (например, DES-MAC, Triple-DES-MAC [PKCS11] или
}  -- HMAC [HMAC, RFC2202])

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

```
 -- Владение подтверждено в этом сообщении (которое содержит
 -- секретный ключ, зашифрованный для )
 subsequentMessage [1] SubsequentMessage,
 -- Владение будет подтверждено в следующем сообщении
 dhMAC             [2] BIT STRING,         -- Отменено
 agreeMAC          [3] PKMACValue,
 encryptedKey      [4] EnvelopedData }

 -- Для keyAgreement (и только для него) владение подтверждено в этом сообщении
 -- (оно содержит MAC (для DER-представления параметра certReq в
 -- CertReqMsg, который ДОЛЖЕН включать subject и publicKey)
 -- на основе ключа, полученного из секретного ключа DH конечного объекта
 -- и открытого ключа DH агентства CA);

SubsequentMessage ::= INTEGER {
 encrCert (0),
 -- Запрос, который приводит к шифрованию сертификата для конечного объекта
 -- (POP будет представлено в подтверждающем сообщении)
 challengeResp (1) }
 -- Запрашивает у CA обмен «запрос-отклик» (challenge-response) с конечным объектом
 -- для подтверждения обладания секретным ключом.

-- Выделение идентификаторов объектов --

-- Регистрационные элементы в CRMF
id-regCtrl OBJECT IDENTIFIER ::= { id-pkip 1 }

id-regCtrl-regToken OBJECT IDENTIFIER ::= { id-regCtrl 1 }
-- с синтаксисом
RegToken ::= UTF8String

id-regCtrl-authenticator OBJECT IDENTIFIER ::= { id-regCtrl 2 }
-- с синтаксисом
Authenticator ::= UTF8String

id-regCtrl-pkiPublicationInfo OBJECT IDENTIFIER ::= { id-regCtrl 3 }
-- с синтаксисом
PKIPublicationInfo ::= SEQUENCE {
action     INTEGER {
             dontPublish (0),
             pleasePublish (1) },
pubInfos  SEQUENCE SIZE (1..MAX) OF SinglePubInfo OPTIONAL }
  -- НЕДОПУСТИМО наличие pubInfos для действия dontPublish
  -- (если указано действие pleasePublish и pubInfos отсутствует,
  -- предполагается dontCare)

SinglePubInfo ::= SEQUENCE {
 pubMethod    INTEGER {
     dontCare    (0),
     x500        (1),
     web         (2),
     ldap        (3) },
 pubLocation  GeneralName OPTIONAL }

id-regCtrl-pkiArchiveOptions     OBJECT IDENTIFIER ::= { id-regCtrl 4 }
-- с синтаксисом
PKIArchiveOptions ::= CHOICE {
 encryptedPrivKey     [0] EncryptedKey,
 -- актуальное значение секретного ключа
 keyGenParameters     [1] KeyGenParameters,
 -- параметры, позволяющие заново создать секретный ключ
 archiveRemGenPrivKey [2] BOOLEAN }
 -- Устанавливается TRUE, если отправитель хочет, чтобы получатель архивировал
 -- секретный ключ пары, которую получатель создал в ответ на этот запрос;
 -- FALSE указывает нежелательность архивирования.

EncryptedKey ::= CHOICE {
 encryptedValue        EncryptedValue,   -- Deprecated
 envelopedData     [0] EnvelopedData }
 -- Зашифрованный секретный ключ ДОЛЖЕН помещаться в строку октетов
 -- envelopedData encryptedContentInfo encryptedContent.

EncryptedValue ::= SEQUENCE {
 intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
 -- Алгоритм, для которого значение будет использоваться
 symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
 -- Симметричный алгоритм, используемый для шифрования значения
 encSymmKey    [2] BIT STRING           OPTIONAL,
 -- (Зашифрованный) симметричный ключ, который будет использоваться для шифрования значения
 keyAlg        [3] AlgorithmIdentifier  OPTIONAL,
 -- Алгоритм, используемый для шифрования симметричного ключа
 valueHint     [4] OCTET STRING         OPTIONAL,
 -- Краткое описание или идентификатор содержимого encValue
 -- (может быть осмысленным только для передающей стороны и применяется лишь
 -- в тех случаях, когда EncryptedValue может быть в будущем перепроверено
 -- передающей стороной)
```

```
 encValue       BIT STRING }
 -- Зашифрованное значение
-- При использовании EncryptedValue для передачи секретного ключа (в отличие от
-- сертификата) реализации ДОЛЖНЫ поддерживать поле encValue, содержащее зашифрованное
-- значение PrivateKeyInfo, как указано в параграфе 12.11 [PKCS11]. Если encValue
-- содержит какой-либо иной формат или кодирование секретного ключа, первый октет
-- valueHint МОЖЕТ служить для указания этого формата или представления (отметим,
-- что возможные значения этого октета ещё не определены). Во всех случаях поле
-- intendedAlg ДОЛЖНО использоваться для указания хотя бы OID предусмотренного алгоритма
-- секретного ключа, если такая информация не известна заранее обеим сторонам.

KeyGenParameters ::= OCTET STRING

id-regCtrl-oldCertID          OBJECT IDENTIFIER ::= { id-regCtrl 5 }
-- с синтаксисом
OldCertId ::= CertId
CertId ::= SEQUENCE {
 issuer           GeneralName,
 serialNumber     INTEGER }

id-regCtrl-protocolEncrKey    OBJECT IDENTIFIER ::= { id-regCtrl 6 }
-- с синтаксисом
ProtocolEncrKey ::= SubjectPublicKeyInfo

-- Регистрационная информация в CRMF
id-regInfo OBJECT IDENTIFIER ::= { id-pkip 2 }

id-regInfo-utf8Pairs    OBJECT IDENTIFIER ::= { id-regInfo 1 }
-- с синтаксисом
UTF8Pairs ::= UTF8String

id-regInfo-certReq       OBJECT IDENTIFIER ::= { id-regInfo 2 }
-- с синтаксисом
CertReq ::= CertRequest

-- id-ct-encKeyWithID является новым типом содержимого для объектов CMS
-- и содержит секретный ключ вместе с идентификатором для агентов депонирования
-- ключей, обеспечивающих возможность проверки при запросах на восстановление.

id-ct-encKeyWithID OBJECT IDENTIFIER ::= {id-ct 21}

EncKeyWithID ::= SEQUENCE {
  privateKey           PrivateKeyInfo,
  identifier CHOICE {
    string             UTF8String,
    generalName        GeneralName
  } OPTIONAL
}

PrivateKeyInfo ::= SEQUENCE {
   version                   INTEGER,
   privateKeyAlgorithm       AlgorithmIdentifier,
   privateKey                OCTET STRING,
   attributes                [0] IMPLICIT Attributes OPTIONAL
}

Attributes ::= SET OF Attribute

END
```

Приложение C. Зачем использовать POP

Процедура подтверждения владения (POP) служит для предоставления удостоверяющему центру (CA) убедительных доказательств того, что сторона, запросившая сертификат для открытого ключа Y, имеет доступ к соответствующему секретному ключу X.

Важность процедуры POP обусловлена тем, что она обеспечивает надлежащий уровень уверенности в корректности работы PKI в целом. На самом низком уровне POP противостоит «самонавязанному отказу в обслуживании» (self-inflicted denial of service), т. е., запрашивающий сам получает сертификат, который не может использоваться для шифрования/расшифровки информации. Однако, как показано в двух примерах ниже, POP препятствует также менее прямым, но более серьёзным угрозам.

POP для ключей подписи. Важно обеспечить POP для ключей, используемых с целью подписывания документов, чтобы обеспечить невозможность отказа от выполненной транзакции. Предположим, например, что Алиса правомерно владеет секретным ключом X, которому соответствует открытый ключ Y. Алиса имеет сертификат от Чарли - УЦ, хранящего Y. Алиса использует ключ X для подписи транзакции T. При отсутствии POP некто Мэл также может получить от Чарли сертификат, содержащий открытый ключ Y. В этом случае возможны две угрозы - Мэл может заявить, что она на самом деле подписала транзакцию T, а Алиса может отказаться от своей подписи T, заявив, что это сделала Мэл. Поскольку нет возможности убедительно доказать или опровергнуть обладание Мэл ключом X, ни одно из указанных выше заявлений не может быть опровергнуто и, следовательно, доверие к PKI теряется (естественно, что при реальном обладании Мэл секретным ключом Алисы X, никакой механизм POP не поможет, но это уже другая проблема).

Отметим, что одним из вариантов защиты может быть включение Алисой (как действительно подписавшей транзакцию) в подписанную информацию своего сертификата или его идентификатора (например, хэш

сертификата). Это может осложнить Мэл претензии на авторство подписи - ей придётся заявить, что она по ошибке поместила сертификат Алисы вместо своего. Однако это не помещает Алисе отказаться от своего авторства подписи. Поскольку сертификат сам по себе является открытым элементом, Мэл действительно могла поместить сертификат Алисы или его идентификатор в подписанную транзакцию и наличие сертификата не является подтверждением того, что Алиса действительно подписала транзакцию, от чего сейчас отказывается. Единственным надёжным способом остановить эту атаку является требование к Мэл доказать обладание ключом X до выдачи его сертификата.

Для ключей подписи, используемых лишь при проверке подлинности, вопрос отказа не возникает и острота этой проблемы снижается, поскольку отказ Алисы после её аутентификации уже не имеет значения и важность POP снижается. Однако и в этом случае **следует** использовать POP по основному или отдельному каналу.

POP для ключей управления ключами. Аналогично, POP для ключей управления ключами (т. е., согласования или обмена ключами) может помочь в предотвращении подрыва доверия к PKI. Предположим, что Эл является новым преподавателем в местном компьютерном университете и создал предварительный вариант экзамена по курсу «Введение в сети», который он читает. Эл хочет передать копию документа декану факультета Дороти для одобрения. Документ зашифрован, поскольку некоторые студенты имеют доступ к компьютерной системе. Однако быстрый поиск в репозитории сертификатов (например, поиск всех записей, где поле subjectPublicKey содержит значение Дороти) показывает, что некоторые студенты пользуются таким же открытом ключом для управления ключами, какой применяет Дороти. В этом случае, если УЦ не использует процедуру POP, Эл не будет иметь возможности узнать, кто из студентов просто создал эти сертификаты, не зная секретного ключа Дороти (это позволяет безопасно передать документ), а кто действительно завладел им (это ставит передачу документа под угрозу расшифровки). Таким образом, услуги PKI, обеспечивающие пользователям уверенность в том, что они общаются именно с тем человеком, который представился, полностью утрачивают доверие. Если УЦ поддерживает процедуру POP, ни у кого из студентов такого сертификата быть не может и это позволяет Элу передать документ Дороти, будучи уверенным в его недоступности студентам.

Адрес автора

Jim Schaad

Soaring Hawk Consulting

PO Box 675

Gold Bar, WA 98251

EMail: jimsch@exmsft.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Полное заявление авторских прав

Интеллектуальная собственность

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.