Network Working Group R. Housley
Request for Comments: 3852 Vigil Security
Obsoletes: 3369 July 2004
Category: Standards Track

Синтаксис CMS

Cryptographic Message Syntax (CMS)

Статус документа

Данный документ содержит спецификацию стандартного протокола Internet, предложенного сообществу Internet, и является приглашением к дискуссии в целях развития этого протокола. Сведения о текущем состоянии стандартизации протокола вы найдете в документе Internet Official Protocol Standards (STD 1). Документ можно распространять без ограничений.

Авторские права

Статус документа

Документ описывает синтаксис криптографических сообщений (CMS[1]), используемый для цифровых подписей, сигнатур, аутентификации и шифрования произвольных содержимого сообщений.

Оглавление

[1]Cryptographic Message Syntax.

1. Введение

Документ описывает синтаксис криптографических сообщений (CMS), используемый для цифровых подписей, дайджестов, проверки подлинности и шифрования произвольного содержимого сообщений.

CMS описывает синтаксис инкапсуляции, служащей для защиты данных. Этот синтаксис поддерживает шифрование и цифровые подписи. Возможна многократная инкапсуляция, когда инкапсулируемые данные уже являются инкапсуляцией других данных. Аналогично, может создаваться цифровая подпись для ранее инкапсулированных данных. Поддерживаются также произвольные атрибуты (например, время создания подписи), которые могут подписываться вместе с сообщением, обеспечивая привязку к подписи других атрибутов, типа заверяющей подписи (countersignature).

CMS может поддерживать множество вариантов архитектуры основанного на сертификатах распространения ключей (таких, как архитектура, разработанная группой PKIX [PROFILE]).

Значения CMS генерируются с применением нотации ASN.1 [X.208-88], использующей представление BER [X.209-88]. Значения обычно представляются в форме строк октетов. Хотя многие системы способны обеспечивать надежную передачу произвольных строк октетов, известно, что многие системы не обеспечивают этого. В этом документе не рассматриваются механизмы кодирования строк октетов для гарантированной передачи в подобных средах.

1.1. Развитие CMS

CMS происходит от PKCS #7 версии 1.5, документированной в RFC 2315 [PKCS#7]. PKCS #7 версии 1.5 была разработана без участия IETF и опубликована изначально как RSA Laboratories Technical Note в ноябре 1993 г. С этого момента ответственность за разработку и поддержку CMS перешла к IETF. В настоящее время несколько важных протоколов, стандартизируемых IETF, используют CMS.

В этом разделе описаны изменения, внесенные IETF в каждую из опубликованных версий CMS.

1.1.1. Отличия от PKCS #7 версии 1.5

RFC 2630 [CMS1] содержал первую версию CMS, включенную в процесс стандартизации IETF. Везде, где это было возможно, обеспечивалась совместимость с PKCS #7 версии 1.5, однако были внесены изменения для согласования с переносом сертификата атрибутов версии 1 и поддержки независимого от алгоритма управления ключами. PKCS #7 версии 1.5 включена только для поддержки транспортировки ключей. В RFC 2630 добавлена поддержка согласования ключей и методы шифрования с использованием заранее распространенных симметричных ключей.

1.1.2. Отличия от RFC 2630

RFC 3369 [CMS2] отменяет действие RFC 2630 [CMS1] и RFC 3211 [PWRI]. Управление ключами на основе паролей включено в спецификацию CMS и задан механизм расширения для поддержки новых схем управления ключами без внесения изменений в CMS. Совместимость со старыми версиями RFC 2630 и RFC 3211 сохранена, однако добавлен перенос сертификатов атрибутов версии 2, а применение сертификатов версии 1 запрещено.

Подписи S/MIME v2 [OLDMSG], основанные на PKCS#7 версии 1.5, совместимы с подписями S/MIME v3 [MSG], основанными на RFC 2630. Однако возникают незначительные проблемы совместимости с подписями на основе PKCS #7 версии 1.5. Эти вопросы рассматриваются в параграфе 5.2.1. Проблемы сохранились и в текущей версии CMS.

Конкретные криптографические алгоритмы не рассматриваются в этом документе, но они рассмотрены в RFC 2630. Обсуждение конкретных криптографических алгоритмов вынесено в отдельный документ [CMSALG]. Такое разделение позволяет IETF независимо обновлять каждый документ. Эта спецификация не требует реализации каких-либо конкретных алгоритмов. Вместо этого в CMS предполагается, что протоколы будут выбирать подходящие алгоритмы для работы в среде протокола. Эти алгоритмы могут выбираться из [CMSALG] или иных документов.

1.1.3. Отличия от RFC 3369

Этот документ отменяет действие RFC 3369 [CMS2]. Как отмечено в предыдущем параграфе, в RFC 3369 добавлен механизм расширения для поддержки схем управления ключами без внесения дополнительных изменений в CMS. Данный документ добавляет похожий механизм расширения для поддержки дополнительных форматов сертификатов

и информации о статусе отзыва без внесения изменений в CMS. Эти расширения документированы в параграфах 10.2.1 и 10.2.2. Сохранена совместимость с предыдущими версиями CMS.

Использование номеров версий описано в параграфе 1.3.

С момента публикации RFC 3369 в документе был обнаружен ряд ошибок, указанных на сайте RFC Editor. В данном документе эти ошибки исправлены.

Прояснен текст параграфа 11.4, описывающего не подписанный атрибут countersignature. Будем надеяться, что новый текст более четко описывает часть подписи SignerInfo, которая покрывается countersignature.

1.2. Терминология

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

1.3. Номера версий

Каждая из основных структур данных включает номер версии в качестве первого элемента структуры. Номера версий предназначены для предотвращения ошибок при декодировании ASN.1. Некоторые реализации не проверяют номер версии перед началом декодирования и только в случае возникновения ошибки просматривают этот номер в процессе обработки ошибочной ситуации. Это разумный подход, поскольку обработка ошибок осуществляется за пределами «быстрого пути». Такой подход также обеспечивает снисходительность к применению отправителями неверного номера версии.

Большинство исходных номеров версий было выделено в рамках PKCS #7 версии 1.5. Другие значения выделялись при определении соответствующих структур данных. При обновлении структур выделялось большее значение номера версии. Однако для обеспечения интероперабельности старший номер версии меняется лишь в случаях изменения синтаксиса. Т. е., выбирается наименьший номер версии, которая поддерживает используемый синтаксис.

2. Общий обзор

Синтаксис CMS является достаточно общим для поддержки разных типов содержимого сообщений. В этом документе для защиты содержимого определяется ContentInfo. Метод ContentInfo инкапсулирует один указанный тип содержимого, а этот тип может поддерживать дополнительную инкапсуляцию. В документе определены 6 типов содержимого — данные (data), подписанные данные (signed-data), вложенные данные (enveloped-data), данные с цифровой подписью (digested-data), шифрованные данные (encrypted-data) и аутентифицированные данные (authenticated-data). Дополнительные типы содержимого могут быть определены в других документах.

Реализация, соответствующая требованиям этого документа, **должна** реализовать ContentInfo, а также **должна** поддерживать типы data, signed-data и enveloped-data. **Могут** поддерживаться также другие типы содержимого.

В качестве общего подхода для каждого типа содержимого возможна однопроходная обработка и использованием представления BER[1] с неопределенным размером. Однопроходные операции особенно хороши для содержимого большого размера, сохраняемого на лентах или получаемого от других процессов через «конвейер» (pipe). Однако такие операции имеют один существенный недостаток — для них сложно выполнить представление с использованием правил DER[2] [X.509-88] в один проход, поскольку размеры различных компонент заранее могут быть не известны. Однако подписанные атрибуты в типе signed-data (подписанные данные) и аутентифицированные атрибуты в типе authenticated-data требуется передавать в формате DER, чтобы получатели могли проверить наличие в содержимом нераспознанных атрибутов. Из числа используемых в CMS атрибутов представление DER требуется только для подписанных и аутентифицированных атрибутов.

3. Базовый синтаксис

Приведенный ниже идентификатор указывает тип информации в содержимом.

```
id-ct-contentInfo OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1) 6 }
```

CMS связывает идентификатор типа содержимого с самим содержимым. Синтаксис **должен** использовать тип ContentInfo в представлении ASN.1.

```
ContentInfo ::= SEQUENCE {
  contentType ContentType,
  content [0] EXPLICIT ANY DEFINED BY contentType }

ContentType ::= OBJECT IDENTIFIER
```

Смысл полей ContentInfo разъяснен ниже.

contentType показывает тип связанного содержимого. Это идентификатор объекта, представляющий собой уникальную строку целых чисел, выделенную агентством, определившим данный тип содержимого.

content представляет собой связанное с объектом содержимое. Тип содержимого может быть уникально определен из contentType. Типы содержимого для data, signed-data, enveloped-data, digested-data, encrypted-data и authenticated-data определены в настоящем документе. При определении в других документах дополнительных типов содержимого определяемому типу ASN.1 **не следует** быть типом CHOICE.

4. Тип содержимого data

Ниже приведен идентификатор типа содержимого data (данные).

[1]Basic Encoding Rules — базовые правила представления.
[2]Distinguished Encoding Rules — правила «изысканного» представления.

```
id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }
```

Этот тип обозначает произвольную строку октетов (например, текст ASCII), интерпретация которой определяется приложением. Такие строки не обязаны иметь внутреннюю структуру (хотя и могут использовать свое определение ASN.1 или иную структуру).

S/MIME использует id-data для идентификации представленного в формате MIME содержимого. Использование этого идентификатора типа определено в RFC 2311 для S/MIME v2 [OLDMSG] и RFC 3851 для S/MIME v3.1 [MSG].

Содержимое типа data обычно инкапсулируется в тип signed-data, enveloped-data, digested-data, encrypted-data или authenticated-data.

5. Тип содержимого Signed-data

Тип signed-data содержит данные любого типа и может включать значения цифровых подписей. Содержимое документа может подписать произвольное число лиц.

Типовым применением типа signed-data являются данные с цифровой подписью содержимого типа data. Другим вариантом применения этого типа является распространение сертификатов и списков отзыва сертификатов (CRL[1]).

Процесс создания типа signed-data включает несколько этапов, перечисленных ниже.

1. Для каждого подписывающего лица рассчитывается значение цифровой подписи или хэш-сумма для содержимого с использованием специфического для данного подписывающего алгоритма. Если подписывается какая-либо информация кроме содержимого, создается цифровая подпись для содержимого и этой информации с использованием алгоритма подписывающего лица (см. параграф 5.4) и результат называется «отпечатком сообщения» (message digest).
2. Для каждого подписывающего отпечаток сообщения подписывается с использованием секретного ключа подписывающего лица.
3. Для каждого подписывающего лица значение подписи и другая специфическая для данного лица информация собираются в значение SignerInfo, как описано в параграфе 5.3. Сертификаты и списки CRL для каждого подписывающего, а также не соответствующие никому из подписавших собираются на этом этапе.
4. Алгоритмы расчета отпечатков для всех подписавших, а также значения SignerInfo для них собираются в значение SignedData, как описано в параграфе 5.1.

Получатель независимо рассчитывает отпечаток сообщения. Полученное значение вместе с открытым ключом подписавшего служат для проверки значения подписи. Открытый ключ подписавшего может быть представлен одним из двух способов. В первом способе ключ представляется названием выпустившей ключ организации вместе с его порядковым номером или идентификатором субъекта ключа, уникально указывающим сертификат, содержащий открытый ключ. В другом варианте ключ может быть представлен идентификатором субъекта, который принимает как сертифицированные, так и не сертифицированные открытые ключи[2]. Хотя это и не требуется, сертификат подписавшего лица может быть включен в поле сертификатов SignedData.

Этот раздел поделен на 6 частей, описывающих типы верхнего уровня SignedData, EncapsulatedContentInfo, SignerInfo для подписавшего, а также расчет отпечатка сообщения, генерацию подписи и ее проверку.

5.1. Тип SignedData

Приведенный ниже идентификатор указывает содержимое типа signed-data.

```
id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
```

Тип signed-data должен иметь форму ASN.1 SignedData

```
SignedData ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos SignerInfos }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

SignerInfos ::= SET OF SignerInfo
```

Значение полей типа SignedData описаны ниже.

version указывает номер версии синтаксиса. Набор допустимых значений определяется полями certificates, eContentType и SignerInfo. Значение version **должно** присваиваться по следующему алгоритму:

```
IF ((имеется поле certificates) AND
   (присутствуют какие-либо сертификаты типа other)) OR
   ((присутствует crls) AND
   (присутствуют любые crls типа other))
THEN version ДОЛЖНО иметь значение 5
ELSE
   IF (имеется поле certificates) AND
```

[1] Certificate revocation list.

[2] В оригинале ошибочно указан только первый способ. См. https://www.rfc-editor.org/errata_search.php?eid=1744. *Прим. перев.*

```
      (присутствуют любые сертификаты атрибутов версии 2)
   THEN version ДОЛЖНО иметь значение 4
   ELSE
      IF ((имеется поле certificates) AND
         ( присутствуют любые атрибуты сертификатов версии 1)) OR
         (все структуры SignerInfo имеют версию 3) OR
         (encapContentInfo eContentType отличается от id-data)
      THEN version ДОЛЖНО иметь значение 3
      ELSE version ДОЛЖНО иметь значение 1
```

digestAlgorithms — набор идентификаторов алгоритмов расчета отпечатка сообщения, который **может** содержать любое (включая 0) число элементов. Каждый элемент определяет алгоритм создания отпечатка (вместе со связанными с ним параметрами), используемого кем-либо из подписавших. Набор содержит список алгоритмов, используемых всеми подписавшими, в произвольном порядке для реализации однопроходной проверки подписей. Разработчики **могут** отказаться от проверки подписей, использующих алгоритмы, не включенные в список. Процесс создания отпечатков сообщений описан в параграфе 5.4.

encapContentInfo — подписанное содержимое, состоящее из идентификатора типа и собственно содержимого. Подробное описание типа EncapsulatedContentInfo приведено в параграфе 5.2.

certificates — набор сертификатов, включающий множество сертификатов, достаточное для поддержки путей сертификации от признанного «корня» (root) или агентства верхнего уровня (top-level certification authority) ко всем подписавшим из поля signerInfos. Набор может содержать избыточные сертификаты, а также может содержать сертификаты для поддержки путей к двум и более независимым агентствам верхнего уровня. Число сертификатов может быть меньше необходимого, если предполагается наличие у получателей других способов получения необходимых сертификатов (например, из предыдущего набора сертификатов). **Могут** включаться сертификаты подписавших. Использование сертификатов атрибутов версии 1 строго запрещено.

crls — набор информации о статусе отзывов. Предполагается, что этих данных предназначено для проверки пригодности сертификатов, указанных в поле certificates, но такое соответствие не является обязательным. Списки CRL являются первичным источником информации о статусе отзыва сертификатов. Поле **может** содержать избыточное число CRL, но **может** включать и недостаточное для проверки число списков.

SignerInfo*s* представляет собой набор информации по каждому подписавшему. Эти данные **могут** содержать произвольное число элементов, включая 0. Подробное описание типа SignerInfo приведено в параграфе 5.3. Поскольку каждый подписавший может применять свой метод цифровой подписи, а будущие спецификации могут менять синтаксис, все реализации **должно** корректно обрабатывать нереализованные версии SignerInfo. Более того, поскольку каждая реализация явно не может поддерживать все возможные алгоритмы цифровой подписи, все реализации **должны** корректно обрабатывать не поддерживаемые алгоритмы цифровой подписи, которые встречаются.

5.2. Тип EncapsulatedContentInfo

Тип EncapsulatedContentInfo имеет структуру

```
EncapsulatedContentInfo ::= SEQUENCE {
  eContentType ContentType,
  eContent [0] EXPLICIT OCTET STRING OPTIONAL }

ContentType ::= OBJECT IDENTIFIER
```

Поля EncapsulatedContentInfo описаны ниже.

eContentType — идентификатор объекта, уникально задающий тип содержимого.

eContent — собственно содержимое, передаваемое в виде строки октетов. Для eContent не требуется представление DER.

Поле eContent может отсутствовать в EncapsulatedContentInfo, что обеспечивает возможность создания «внешних подписей». В случае внешней подписи подписываемое содержимое отсутствует в EncapsulatedContentInfo, включаемом в тип signed-data. Если значение eContent отсутствует в EncapsulatedContentInfo, signatureValue рассчитывается и значение eContentType назначается как при наличии значения eContent.

В вырожденном случае отсутствия подписавших лиц значение EncapsulatedContentInfo не будет относиться к делу. В такой ситуации «подписываемый» тип содержимого в EncapsulatedContentInfo **должен** быть id-data (см. раздел 4), а поле содержимого в EncapsulatedContentInfo **должно** быть опущено.

5.2.1. Совместимость с PKCS #7

В этом параграфе приводится предупреждение для разработчиков, желающих поддерживать SignedData для типов данных CMS и PKCS #7 [PKCS#7] одновременно. Оба типа CMS и PKCS #7 указывают инкапсулированное содержимое с идентификатором объекта, но тип ASN.1 самого содержимого является переменным для PKCS #7 SignedData.

PKCS #7 определяет content как

```
content [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL
```

CMS определяет eContent как

```
eContent [0] EXPLICIT OCTET STRING OPTIONAL
```

Определение CMS значительно проще в использовании для большинства приложений и совместимо с обоими форматами S/MIME v2 и S/MIME v3. Подписанные сообщения S/MIME, использующие CMS и PKCS #7, совместимы, поскольку форматы подписанных сообщений заданы идентично в RFC 2311 для S/MIME v2 [OLDMSG] и RFC 3851 для S/MIME v3.1 [MSG]. S/MIME v2 инкапсулирует содержимое MIME в тип Data (строка октетов - OCTET STRING), передаваемые в SignedData contentInfo **любого** поля, а S/MIME v3 передает содержимое MIME в строке октетов SignedData encapContentInfo eContent. Следовательно, в S/MIME v2 и S/MIME v3 содержимое MIME помещается в

OCTET STRING и отпечаток сообщения рассчитывается для идентичных частей содержимого. Т. е., отпечаток сообщения рассчитывается по октетам OCTET STRING без учета октетов тегов и размера.

Между типами CMS и PKCS #7 для SignedData имеется несовместимость, когда инкапсулированное содержимое не форматировано с использованием типа Data. Например, когда подписанная с использованием RFC 2634 [ESS] информация инкапсулируется в тип CMS SignedData, последовательность Receipt SEQUENCE представляется строкой октетов SignedData encapContentInfo eContent, а отпечаток сообщения рассчитывается с использованием всей последовательности Receipt (включая октеты тега, размера и значения). Однако, если подписанное в соответствии с RFC 2634 содержимое инкапсулируется в тип PKCS #7 SignedData, последовательность Receipt представляется в формате DER [X.509-88] SignedData contentInfo **любого** поля (SEQUENCE не является OCTET STRING). Следовательно, отпечаток сообщения рассчитывается только для октетов последовательности Receipt.

Показанный ниже метод может использоваться для обеспечения совместимости с PKCS #7 при обработке содержимого типа SignedData. Если реализаций не способна выполнить декодирование ASN.1 для типа SignedData, использующего синтаксис строки октетов CMS SignedData encapContentInfo eContent, она **может** попытаться декодировать тип SignedData используя синтаксис PKCS #7 SignedData contentInfo и рассчитывая соответствующим образом подпись сообщения.

Приведенный ниже метод может использоваться для обеспечения совместимости с PKCS #7 при создании содержимого типа SignedData, в котором инкапсулируемые данные не форматированы с применением типа Data. Реализация **может** проверить значение eContentType и подправить ожидаемое DER-представление eContent на основе значения идентификатора объекта. Например, для поддержки Microsoft Authenticode [MSAC] **можно** выполнить следующее:

```
eContentType Object Identifier устанавливается в { 1 3 6 1 4 1 311 2 1 4 }
eContent содержит информацию Authenticode в представлении DER.
```

5.3. Тип SignerInfo

Информация для каждого подписавшего представляется типом SignerInfo.

```
SignerInfo ::= SEQUENCE {
  version CMSVersion,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier }

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
  attrType OBJECT IDENTIFIER,
  attrValues SET OF AttributeValue }

AttributeValue ::= ANY

SignatureValue ::= OCTET STRING
```

Поля SignerInfo описаны ниже.

version — номер версии синтаксиса. Если в SignerIdentifier используется выбор (CHOICE) issuerAndSerialNumber, для номера версии **должно** устанавливаться значение 1. Если SignerIdentifier представляет собой subjectKeyIdentifier, **должен** устанавливаться номер версии 3.

sid указывает сертификат (и, следовательно, открытый ключ) подписавшего. Открытый ключ подписавшего требуется получателю для проверки подписи. SignerIdentifier обеспечивает два варианта указания открытого ключа подписавшего. Вариант issuerAndSerialNumber идентифицирует ключ по имени эмитента и порядковому номеру сертификата, вариант subjectKeyIdentifier — по идентификатору ключа. При указании сертификата X.509 идентификатор ключа соответствует значению расширения X.509 subjectKeyIdentifier. При указании других форматов документ, задающий формат и использование сертификата с CMS, должен включать детали идентификатора соответствующего ключа в подходящем поле сертификата. Реализации **должны** поддерживать восприятие форм issuerAndSerialNumber и subjectKeyIdentifier для SignerIdentifier. При генерации SignerIdentifier реализация **может** поддерживать лишь одну из этих форм (issuerAndSerialNumber или subjectKeyIdentifier) и всегда применять ее, а **может** также произвольно смешивать применение этих двух форм. Однако **должен** применяться идентификатор subjectKeyIdentifier для указания открытого ключа, содержащегося в сертификате, отличном от X.509.

digestAlgorithm идентифицирует алгоритм цифровой подписи сообщения и все связанные с ними параметры, использованные подписавшим. Отпечаток (подпись) сообщения рассчитывается для подписываемого содержимого или для содержимого вместе с атрибутами подписи, как описано в параграфе 5.4. Алгоритм подписи сообщения **следует** указывать в поле digestAlgorithms связанных с ним данных SignerData. Реализации **могут** отказываться от проверки подписей, использующих не включенный в SignedData digestAlgorithms алгоритм.

SignedAttrs представляет набор подписанных атрибутов. Поле является необязательным, но оно **должно** присутствовать, если подписываемое значение EncapsulatedContentInfo не является id-data. SignedAttributes **должно** представляться в формате DER даже при использовании для остальной структуры представления BER. Полезные типы атрибутов (такие, как время подписания) определены в разделе 11. Если данное поле присутствует, оно **должно** содержать по крайней мере два приведенных ниже атрибута:

content-type в качестве своего значения использует тип содержимого EncapsulatedContentInfo подписываемого значения. Атрибут content-type описан в параграфе 11.1. Однако атрибут content-type **недопустимо** применять как часть не подписанного атрибута countersignature, определенного в параграфе 11.4.

message-digest в качестве своего значения использует отпечаток сообщения для содержимого. Атрибут message-digest определен в параграфе 11.2.

signatureAlgorithm указывает алгоритм подписи и все связанные с ним параметры, использованные подписывающим при создании цифровой подписи.

signature — результат генерации цифровой подписи с использованием отпечатка сообщения и секретного ключа подписывающего. Детали подписи зависят от использованного алгоритма.

unsignedAttrs — набор атрибутов, которые не были подписаны. Это поле является необязательным. Полезные типы атрибутов (например, countersignatures) описаны в разделе 11.

Значения полей типов SignedAttribute и UnsignedAttribute описаны ниже.

attrType указывает тип атрибута и является идентификатором объекта.

attrValues указывает множество значений, составляющих атрибут. Тип каждого значения в этом множестве уникально определяется полем attrType (поле attrType может накладывать ограничения на число элементов множества).

5.4. Процесс расчета отпечатка сообщения

В процессе расчета отпечатка сообщения учитывается лишь само подписываемое сообщение или сообщение вместе с подписанными атрибутами. В любом случае исходной информацией для процесса расчета отпечатка является «значение» инкапсулируемого содержимого, которое будет подписано. В частности, исходной информацией служит строка октетов (OCTET STRING) encapContentInfo eContent, к которой применяется процесс подписания. Для алгоритма цифровой подписи сообщения входной информацией служат только строка октетов eContent без тегов и поля размера.

Результат процесса расчета отпечатка сообщения зависит от наличия поля signedAttrs. В отсутствие этого поля результатом будет просто отпечаток сообщения, как описано выше. Если же поле имеется, результатом будет отпечаток полного DER-представления значения SignedAttrs, содержащегося в поле signedAttrs. Поскольку значение SignedAttrs (при его наличии) должно содержать атрибуты content-type и message-digest, их значения также опосредованно включаются в результат. Атрибут content-type **недопустимо** включать в неподписанный атрибут countersignature, как описано в параграфе 11.4. При расчете отпечатка выполняется раздельное представление поля signedAttrs. Тег IMPLICIT [0] в signedAttrs не используется для представления DER и взамен применяется тег EXPLICIT SET OF. Т. е. в расчет **должно** включаться DER-представление тега EXPLICIT SET OF (а не IMPLICIT [0]) вместе с октетами размера и содержимого SignedAttributes.

При отсутствии поля signedAttrs для расчета отпечатка сообщения используется только строка октетов SignedData encapContentInfo eContent (например, содержимое файла). Преимущество этого заключается в том, что не требуется заранее знать размер подписываемого содержимого при генерации подписи.

Хотя строка октетов тега encapContentInfo eContent не включается в расчет отпечатка сообщения, ее можно защитить иными средствами. Октеты размера защищаются самой природой алгоритма цифровой подписи, поскольку практически невозможно найти два разных сообщения любого размера, которые будут давать одинаковые отпечатки.

5.5. Процесс создания подписи

Входные данные для процесса генерации подписи включают результат процесса создания отпечатка сообщения (message digest) и секретный ключ подписывающего. Детали процесса генерации подписи зависят от применяемого алгоритма. Идентификатор объекта вместе со всеми параметрами, задающими алгоритм, используемый подписывающим, передаются в поле signatureAlgorithm. Значение подписи, созданное подписывающим, **должно** представляться в форме строки октетов (OCTET STRING) и передаваться в поле signature.

5.6. Процесс проверки подписи

Входные данные для процесса проверки подписи включают результат процесса расчета отпечатка сообщения и открытый ключ подписавшего. Получатель **может** получить корректный открытый ключ подписавшего разными способами, но предпочтительно использовать метод из сертификата в поле SignedData certificates. Выбор и проверка пригодности ключа подписавшего **могут** базироваться на проверке пути сертификации (см. [PROFILE]), а также на иных способах, но этот вопрос выходит за рамки данного документа. Детали проверки подписи зависят от использованного алгоритма.

Получателю **недопустимо** полагаться на какие-либо значения отпечатка сообщения, рассчитанные его создателем. Если поле SignedData signerInfo включает signedAttributes, отпечаток содержимого сообщения **должен** быть рассчитан в соответствии с параграфом 5.4. Для того, чтобы подпись считалась действительной, рассчитанный отпечаток **должен** совпасть со значением атрибута messageDigest, включенным в поле signedAttributes структуры SignedData signerInfo.

Если SignedData signerInfo включает signedAttributes, значение атрибута content-type **должно** совпадать со значением SignedData encapContentInfo eContentType.

6. Тип содержимого Enveloped-data

Содержимое типа enveloped-data состоит из зашифрованных данных любого типа и зашифрованных ключей шифрования содержимого (encrypted content-encryption key) для одного или множества получателей. Комбинация шифрованного содержимого и шифрованного ключа шифрования содержимого для получателя представляет собой «цифровой конверт» (digital envelope) для данного получателя. Содержимое любого типа может быть упаковано в конверты для произвольного числа получателей с использованием любого из поддерживаемых методов управления ключами для каждого из получателей.

Типовым применением содержимого типа enveloped-data служит представление «цифровых конвертов» для одного или множества получателей с содержимым типа data или signed-data.

Этапы создания цифрового конверта Enveloped-data перечислены ниже.

1. Ключи шифрования содержимого для конкретного алгоритма генерируются случайным образом.
2. Ключи шифрования содержимого шифруются для каждого получателя. Детали этого шифрования зависят от применяемого механизма управления ключами и поддерживаются четыре основных метода:

   key transport (доставка ключей) — ключ шифрования содержимого шифруется с помощью открытого ключа получателя;

   key agreement (согласование ключей) — используются открытый ключ получателя и секретный ключ отправителя для генерации симметричного ключа, который применяется для шифрования ключей шифрования содержимого;

   symmetric key-encryption keys (шифрование с симметричным ключом) — ключ шифрования содержимого шифруется с использованием заранее полученного симметричного ключа, известного и получателю;

   passwords (пароль) — ключ шифрования содержимого шифруется с помощью ключа, генерируемого из пароля и некого заранее известного сторонам секретного значения.
3. Для каждого получателя зашифрованный ключ шифрования содержимого и другая информация для конкретного получателя собираются в значение RecipientInfo, как описано в параграфе 6.2.
4. Содержимое шифруется с использованием ключа content-encryption. При этом в зависимости от применяемого ключа и алгоритма может потребоваться дополнение шифруемого содержимого для выравнивания по размеру блока шифрования (см. параграф 6.3).
5. Значения RecipientInfo для всех получателей собираются вместе с зашифрованным содержимым в структуру EnvelopedData, описанную в параграфе 6.1.

Получатель открывает «цифровой конверт», расшифровывая один из ключей шифрования содержимого, с помощью которого далее расшифровывается само содержимое.

Далее рассматривается структура верхнего уровня EnvelopedData, а затем тип RecipientInfo с информацией для конкретного получателя, а также процессы расшифровки ключей шифрования содержимого и самого содержимого.

6.1. Тип EnvelopedData

Тип содержимого enveloped-data указывает идентификатор

```
id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }
```

Тип enveloped-data должен иметь форму ASN.1 EnvelopedData

```
EnvelopedData ::= SEQUENCE {
 version CMSVersion,
 originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
 recipientInfos RecipientInfos,
 encryptedContentInfo EncryptedContentInfo,
 unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }

OriginatorInfo ::= SEQUENCE {
 certs [0] IMPLICIT CertificateSet OPTIONAL,
 crls [1] IMPLICIT RevocationInfoChoices OPTIONAL }

RecipientInfos ::= SET SIZE (1..MAX) OF RecipientInfo

EncryptedContentInfo ::= SEQUENCE {
 contentType ContentType,
 contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
 encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }

EncryptedContent ::= OCTET STRING

UnprotectedAttributes ::= SET SIZE (1..MAX) OF Attribute
```

Значения полей EnvelopedData описаны ниже.

version — номер версии синтаксиса. Значение зависит от полей originatorInfo, RecipientInfo и unprotectedAttrs и **должно** присваиваться по следующему алгоритму:

```
IF (присутствует originatorInfo) AND
   ((присутствуют любые сертификаты типа other) OR
   (присутствуют любые crls типа other))
```

```
        THEN устанавливается version = 4
        ELSE
           IF ((присутствует originatorInfo) AND
              (присутствуют любые сертификаты атрибутов версии 2)) OR
              (любая из структур RecipientInfo включает pwri) OR
              (любая из структур RecipientInfo включает ori)
           THEN устанавливается version = 3
           ELSE
              IF (originatorInfo отсутствует) AND[1]
                 (unprotectedAttrs отсутствует) AND[1]
                 (все структуры RecipientInfo имеют версию 0)
              THEN устанавливается version = 0
              ELSE устанавливается version = 2
```

originatorInfo — необязательное поле с информацией инициаторе; поле включается только в тех случаях, когда этого требует алгоритм управления ключами. Поле может содержать сертификаты и списки CRL:

> ***certs*** — набор сертификатов, который может содержать сертификаты инициатора, связанные с несколькими разными алгоритмами управления ключами. Поле certs может также содержать сертификаты атрибутов, связанные с инициатором. Сертификаты в certs предназначены для того, чтобы обеспечить всем получателям возможность построения пути от признаваемого корня (root) или удостоверяющего центра верхнего уровня (top-level certification authority). Однако certs может содержать избыточные сертификаты, позволяющие построить несколько путей для двух и более удостоверяющих центров. Возможна и нехватка сертификатов в поле certs, если у получателя имеются другие способы получения требуемых сертификатов (например, предшествующий набор сертификатов).
>
> ***crls*** — набор списков отзыва сертификатов (CRL). Это поле содержит информацию, которой может быть достаточно для проверки действительности сертификатов, указанных в поле certs, но эта достаточность не требуется. Это поле **может** содержать избыточное или недостаточное число списков CRL.

recipientInfos — набор информации, предназначенной для конкретного получателя. В наборе **должен** присутствовать по крайней мере один элемент.

encryptedContentInfo — зашифрованное содержимое.

unprotectedAttrs — необязательный набор незашифрованных атрибутов. Полезные типы атрибутов определены в разделе 11.

Поля типа EncryptedContentInfo имеют приведенные ниже значения

contentType указывает тип содержимого.

contentEncryptionAlgorithm указывает алгоритм шифрования содержимого и все связанные с ним параметры, которые применяются при шифровании содержимого. Процесс шифрования содержимого описан в параграфе 6.3. Для всех получателей используются одни и те же алгоритм и ключ шифрования содержимого.

encryptedContent — результат шифрования содержимого. Это поле является необязательным и при его отсутствии шифрованное содержимое должно доставляться иным способом.

Поле recipientInfos размещается перед полем encryptedContentInfo, что позволяет обрабатывать значение EnvelopedData за один проход.

6.2. Тип RecipientInfo

Предназначенная для отдельного получателя информация представляется типом RecipientInfo. Формат RecipientInfo отличается для каждого из поддерживаемых методов управления ключами. Для каждого получателя одного зашифрованного содержимого можно применять любой из методов управления ключами. Во всех случаях зашифрованный ключ шифрования содержимого передается одному или множеству получателей.

Поскольку не каждая реализация будет поддерживать все возможные алгоритмы управления ключами, все реализации **должны** корректно обрабатывать нереализованные алгоритмы, которые могут встречаться. Например, если получателю приходит ключ шифрования содержимого, зашифрованный с использованием его открытого ключа RSA и RSA-OAEP, а реализация поддерживает только RSA PKCS #1 v1.5, в ней должна быть реализована процедура корректного отказа.

Реализации **должны** поддерживать доставку ключей (key transport), согласование ключей (key agreement) и применение известных симметричных ключей, представленные ktri, kari и kekri, соответственно. Реализации **могут** поддерживать управление ключами на основе пароля, представляемое pwri. Реализации также **могут** поддерживать любой другой метод управления ключами, представляемый ori. Поскольку каждый получатель может использовать свой метод управления ключами, а будущие спецификации могут добавлять новые методы, все реализации **должны** корректно обрабатывать не реализованные в них варианты RecipientInfo CHOICE и **должны** корректно обрабатывать не реализованные версии поддерживаемых вариантов CHOICE, а также **должны** корректно обрабатывать неизвестные или не реализованные варианты ori.

```
RecipientInfo ::= CHOICE {
  ktri KeyTransRecipientInfo,
  kari [1] KeyAgreeRecipientInfo,
  kekri [2] KEKRecipientInfo,
  pwri [3] PasswordRecipientinfo,
  ori [4] OtherRecipientInfo }

EncryptedKey ::= OCTET STRING
```

[1] В оригинале ошибочно сказано OR. См. https://www.rfc-editor.org/errata_search.php?eid=222. *Прим. перев.*

6.2.1. KeyTransRecipientInfo

Информация для получателей, использующих доставку ключей (key transport), представляется типом KeyTransRecipientInfo. Каждый экземпляр KeyTransRecipientInfo переносит ключ шифрования содержимого для одного получателя.

```
KeyTransRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- всегда имеет значение 0 или 2
  rid RecipientIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey EncryptedKey }

RecipientIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier }
```

Поля типа KeyTransRecipientInfo описаны ниже.

version — номер версии синтаксиса. Если RecipientIdentifier представляет собой CHOICE issuerAndSerialNumber, номер версии **должен** иметь значение 0. Если RecipientIdentifier представляет собой subjectKeyIdentifier, номер версии **должен** быть 2.

rid указывает сертификат или ключ получателя, который отправитель применил для защиты ключа шифрования содержимого путем шифрования с помощью открытого ключа получателя. RecipientIdentifier поддерживает два варианта указания сертификата и, соответственно, открытого ключа получателя. Сертификат получателя должен содержать открытый ключ для транспортировки ключей. Следовательно, сертификат получателя X.509 версии 3, содержащий расширение использования ключей (key usage), **должен** иметь флаг keyEncipherment. Вариант issuerAndSerialNumber указывает сертификат получателя по отличительному имени эмитента и номеру сертификата, а subjectKeyIdentifier — по идентификатору ключа. При указании сертификата X.509 идентификатор ключа соответствует значению расширения X.509 subjectKeyIdentifier. При указании сертификатов иного формата документ, задающий формат сертификата и его использование с CMS, должен включать детали сопоставления идентификатора ключа с подходящим полем сертификата. На стороне получателя реализации **должны** поддерживать оба варианта указания сертификата получателя, на стороне отправителя **должен** поддерживаться хотя бы один из этих вариантов.

keyEncryptionAlgorithm указывает алгоритм шифрования ключа и все связанные с ним параметры, которые потребуются получателю для расшифровки ключа шифрования содержимого. Процесс шифрования ключей описан в параграфе 6.4.

encryptedKey - результат шифрования ключа шифрования содержимого для данного получателя.

6.2.2. KeyAgreeRecipientInfo

Информация для получателя, использующего согласование ключей (key agreement), представляется типом KeyAgreeRecipientInfo. Каждый экземпляр KeyAgreeRecipientInfo будет содержать ключ шифрования содержимого для одного или множества получателей использующих один алгоритм согласования ключей и общие параметры для него.

```
KeyAgreeRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- всегда устанавливается значение 3
  originator [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys }

OriginatorIdentifierOrKey ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier,
  originatorKey [1] OriginatorPublicKey }

OriginatorPublicKey ::= SEQUENCE {
  algorithm AlgorithmIdentifier,
  publicKey BIT STRING }

RecipientEncryptedKeys ::= SEQUENCE OF RecipientEncryptedKey

RecipientEncryptedKey ::= SEQUENCE {
  rid KeyAgreeRecipientIdentifier,
  encryptedKey EncryptedKey }

KeyAgreeRecipientIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  rKeyId [0] IMPLICIT RecipientKeyIdentifier }

RecipientKeyIdentifier ::= SEQUENCE {
  subjectKeyIdentifier SubjectKeyIdentifier,
  date GeneralizedTime OPTIONAL,
  other OtherKeyAttribute OPTIONAL }

SubjectKeyIdentifier ::= OCTET STRING
```

Поля типа KeyAgreeRecipientInfo описаны ниже.

version — номер версии синтаксиса. Всегда **должен** иметь значение 3.

originator представляет собой выбор (CHOICE) из трех вариантов указания открытого ключа отправителя для согласования ключей. Отправитель использует соответствующий секретный ключ и открытый ключ получателя для генерации парного ключа, с помощью которого шифруется ключ шифрования содержимого. Вариант issuerAndSerialNumber задает сертификат отправителя и, следовательно, его открытый ключ путем указания отличительного имени эмитента сертификата и порядкового номера сертификата. Вариант subjectKeyIdentifier указывает сертификат и ключ отправителя с помощью идентификатора ключа. Для сертификата X.509 идентификатор ключа соответствует значению расширения X.509 subjectKeyIdentifier. Для других форматов сертификата документ со спецификацией этого формата и его применения в CMS должен подробно описывать сопоставление идентификатора подходящему полю сертификата. Вариант originatorKey указывает идентификатор алгоритма и открытый ключ отправителя для согласования ключей. Этот метод обеспечивает анонимность инициатора, поскольку открытый ключ не сертифицируется. Реализации **должны** поддерживать все три варианта указания открытого ключа отправителя.

ukm является не обязательным. Для некоторых алгоритмов согласования ключей отправитель представляет ключевой материал UKM[1], чтобы гарантировать создание нового ключа каждый раз при взаимодействии с тем же партнером. Реализации **должны** воспринимать последовательности ***KeyAgreeRecipientInfo***, включающие поле ukm. Реализации, не поддерживающие алгоритмы согласования ключей с использованием UKM, **должны** аккуратно обрабатывать наличие UKM.

keyEncryptionAlgorithm указывает алгоритм шифрования ключа и все связанные с ним параметры, использованные для зашифровки ключа шифрования содержимого с помощью этого ключа. Процесс шифрования ключей описан в параграфе 6.4.

recipientEncryptedKeys включает идентификатор получателя и зашифрованный ключ для одного или множества получателей. KeyAgreeRecipientIdentifier представляет собой выбор (CHOICE) из двух вариантов задания сертификата получателя и, следовательно, его открытого ключа, который был использован отправителем для генерации парного ключа для зашифровки ключа шифрования содержимого. Сертификат получателя должен содержать открытый ключ, применяемый для согласования ключей. Следовательно, сертификат получателя X.509 версии 3 с расширением key usage **должен** иметь установленный бит keyAgreement. Ключ шифрования содержимого шифруется с использованием парного ключа. Вариант issuerAndSerialNumber указывает сертификат получателя по отличительному имени эмитента и порядковому номеру сертификата, тип RecipientKeyIdentifier описан ниже. encryptedKey представляет собой результат зашифровки ключа шифрования содержимого с помощью парного ключа, генерируемого с использованием алгоритма согласования ключей. Реализации **должны** поддерживать оба варианта указания сертификата получателя.

Поля типа RecipientKeyIdentifier описаны ниже.

subjectKeyIdentifier указывает сертификат получателя идентификатором ключа. При указании сертификата X.509 идентификатор ключа соответствует значению расширения X.509 subjectKeyIdentifier. Для других сертификатов документы со спецификацией формата и использования с CMS должны включать детали сопоставления идентификатора ключа подходящему полю сертификата.

date — необязательное поле, которое указывает получателю дату предшествующего распространения ключевого материала UKM, который был использован отправителем.

other — необязательное поле, содержащее дополнительную информацию, которую получатель применяет для нахождения открытого ключевого материала, использованного получателем.

6.2.3. KEKRecipientInfo

Информация для получателей, использующих заранее известный симметричный ключ, представляется типом KEKRecipientInfo. Каждый экземпляр KEKRecipientInfo будет включать ключ шифрования содержимого для одного или множества получателей, зашифрованный с использованием заранее распространенного симметричного ключа.

```
KEKRecipientInfo ::= SEQUENCE {
  version CMSVersion,  -- всегда устанавливается значение 4
  kekid KEKIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey EncryptedKey }

KEKIdentifier ::= SEQUENCE {
  keyIdentifier OCTET STRING,
  date GeneralizedTime OPTIONAL,
  other OtherKeyAttribute OPTIONAL }
```

Поля типа KEKRecipientInfo описаны ниже

version — номер версии синтаксиса. Всегда **должен** иметь значение 4.

kekid указывает симметричный ключ, которым зашифрован ключ шифрования содержимого, заранее переданнный отправителю и одному или множеству получателей.

keyEncryptionAlgorithm указывает алгоритм шифрования ключа и все связанные с ним параметры, которые нужны для расшифровки ключа шифрования содержимого. Процесс шифрования ключей описан в параграфе 6.4.

encryptedKey — результат зашифровки ключа шифрования содержимого с помощью симметричного ключа.

Поля типа KEKIdentifier описаны ниже.

keyIdentifier — указывает ключ шифрования ключей, который был независимо передан отправителю и одному или множеству получателей.

[1]User Keying Material — пользовательский ключевой материал.

date — необязательное поле, которое может указывать дату, идентифицирующую один ключ шифрования ключей из заранее распространенного набора таких ключей.

other — необязательное поле, которое может содержать дополнительную информацию, позволяющую получателю определить использованный отправителем ключ шифрования ключа.

6.2.4. PasswordRecipientInfo

Информация для получателей, использующих пароль или общий секрет, представляется типом PasswordRecipientInfo. Каждый экземпляр PasswordRecipientInfo будет передавать ключ шифрования содержимого для одного или множества получателей, которые знают пароль или общий секрет.

Тип PasswordRecipientInfo задан в RFC 3211 [PWRI]. Структура PasswordRecipientInfo приведена здесь лишь для полноты.

```
PasswordRecipientInfo ::= SEQUENCE {
  version CMSVersion,   -- всегда устанавливается значение 0
  keyDerivationAlgorithm [0] KeyDerivationAlgorithmIdentifier OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier, encryptedKey EncryptedKey }
```

Поля типа PasswordRecipientInfo описаны ниже.

version — номер версии синтаксиса. Всегда **должен** иметь значение 0.

keyDerivationAlgorithm указывает алгоритм создания ключа и все связанные с ним параметры, которые нужны для воспроизведения знающим пароль или общий секрет получателем зашифрованного ключа шифрования содержимого. Отсутствие этого поля означает передачу ключа шифрования ключа иным способом (например, с помощью смарт-карты или крипто-токена).

keyEncryptionAlgorithm указывает алгоритм шифрования и все связанные с ним параметры, использованные для зашифровки ключа шифрования содержимого с помощью ключа шифрования ключей.

encryptedKey — результат зашифровки ключа шифрования содержимого с помощью ключа шифрования ключей.

6.2.5. OtherRecipientInfo

Информация для получателей, использующих иные методы управления ключами, представляется типом OtherRecipientInfo. Этот тип позволяет использовать другие методы, в дополнение к транспортировке ключей, их согласованию, применению известных симметричных ключей, управлению ключами на основе паролей, а также методы, которые будут определены в будущих документах. Ниже показан формат идентификатора объекта, уникально указывающего такие методы управления ключами.

```
OtherRecipientInfo ::= SEQUENCE {
  oriType OBJECT IDENTIFIER,
  oriValue ANY DEFINED BY oriType }
```

Поля типа OtherRecipientInfo описаны ниже.

oriType указывает метод управления ключами.

oriValue содержит элементы протокольных данных, требуемые получателю, который использует указанный метод управления ключами.

6.3. Процесс шифрования содержимого

Случайным образом генерируется ключ шифрования содержимого для нужного алгоритма. Защищаемые данные дополняются в соответствии с приведенным ниже описанием и после этого шифруются с использованием созданного ключа. Операция шифрования отображает произвольную строку октетов (исходные данные) в другую строку октетов (шифрованные данные) с использованием ключа шифрования содержимого. Зашифрованные данные включаются в строку октетов EnvelopedData encryptedContentInfo encryptedContent.

В некоторых алгоритмах шифрования предполагается, что размер исходных данных кратен некому целому числу k, где k больше 1. Для таких алгоритмов к входным данным в конце будут добавляться октеты заполнения со значениями k - (lth mod k) в количестве k - (lth mod k), где lth - размер исходных данных. Иными словами, после исходных данных будут добавляться последовательности

```
                01 -- если lth mod k = k-1
             02 02 -- если lth mod k = k-2
                .
                .
                .
      k k ... k k -- если lth mod k = 0
```

Октеты заполнения могут быть однозначно идентифицированы и удалены, включая ситуацию, когда исходные данные имеют размер, кратный величине блока и заполнения не используется. Этот метод заполнения корректно работает только в тех случаях, когда размер блока (k) меньше 256.

6.4. Процесс шифрования ключа

Исходными данными для процесса шифрования ключа (значение, представляемое алгоритму шифрования ключей у получателя) является просто «значение» ключа шифрования содержимого.

Для каждого получателя общего набора зашифрованной информации может применяться любой из упомянутых выше методов управления ключами.

7. Тип содержимого Digested-data

Тип содержимого digested-data включает данные любого типа с отпечатком (message digest) для них.

Обычно тип digested-data используется для защиты целостности содержимого и результат процесса создания подписанных данных служит входной информацией для создания типа enveloped-data.

Создание типа digested-data включает два этапа:

1. рассчитывается отпечаток (хэш) содержимого с использованием алгоритма message-digest;
2. алгоритм message-digest и отпечаток сообщения вместе с подписанным содержимым объединяются в структуру DigestedData.

Получатель проверяет отпечаток сообщения, сравнивая полученное значение с рассчитанным им самим отпечатком.

Ниже приведен идентификатор объекта для содержимого типа digested-data.

```
id-digestedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 5 }
```

Тип digested-data должен иметь форму ASN.1 DigestedData:

```
DigestedData ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithm DigestAlgorithmIdentifier,
  encapContentInfo EncapsulatedContentInfo,
  digest Digest }

Digest ::= OCTET STRING
```

Поля типа DigestedData описаны ниже.

version - номер версии синтаксиса. Если инкапсулировано содержимое типа id-data, для номера версии **должно** быть указано значение 0, однако для остальных типов **должно** устанавливаться значение 2.

digestAlgorithm — указывает алгоритм цифровой подписи и все связанные с ним параметры, использованные для создания подписи. Процесс создания цифровой подписи совпадает с описанным в параграфе 5.4, если не подписывается дополнительных атрибутов.

encapContentInfo — подписываемое содержимое, как определено в параграфе 5.2.

digest — результат процесса создания подписи для сообщения.

Порядок размещения полей digestAlgorithm, encapContentInfo и digest делает возможной обработку значения DigestedData за один проход.

8. Тип содержимого Encrypted-data

Тип encrypted-data содержит зашифрованные данные любого типа. В отличие от enveloped-data тип encrypted-data не включает ни получателей, ни зашифрованных ключей шифрования содержимого. Управление ключами **должно** осуществляться иным способом.

Типовым применением encrypted-data является шифрование содержимого для локального хранения (возможно, с генерацией ключа шифрования по паролю).

Ниже показан идентификатор объекта для содержимого типа encrypted-data.

```
id-encryptedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 6 }
```

Тип encrypted-data должен иметь форму ASN.1 EncryptedData

```
EncryptedData ::= SEQUENCE {
  version CMSVersion,
  encryptedContentInfo EncryptedContentInfo,
  unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }
```

Поля типа EncryptedData описаны ниже

version - номер версии синтаксиса. При наличии unprotectedAttrs номер версии **должен** иметь значение 2, при отсутствии unprotectedAttrs **должно** указываться значение 0.

encryptedContentInfo — зашифрованное содержимое, как описано в параграфе 6.1.

unprotectedAttrs — набор атрибутов, которые не шифруются. Это поле не является обязательным. Полезные типы атрибутов определены в разделе 11.

9. Тип содержимого Authenticated-data

Тип authenticated-data включает содержимое произвольного типа, код аутентификации сообщения (MAC[1]) и зашифрованные ключи аутентификации для одного или множества получателей. Комбинация MAC и одного из зашифрованных ключей аутентификации требуется получателю для проверки целостности содержимого. Защита целостности может обеспечиваться для любого типа содержимого и произвольного числа получателей.

Процесс создания authenticated-data включает перечисленные ниже шаги.

1. Случайным образом генерируется ключ аутентификации сообщения для конкретного алгоритма аутентификации.

[1]Message authentication code.

2. Ключ аутентификации сообщения шифруется для каждого получателя. Детали этого шифрования зависят от используемого алгоритма управления ключами.

3. Для каждого получателя зашифрованный ключ аутентификации сообщения вместе с другой, относящейся к этому получателю, информацией собираются в значение RecipientInfo, как описано в параграфе 6.2.

4. Используя ключ аутентификации сообщения, инициатор рассчитывает значение MAC для содержимого. Если инициатор в дополнение к содержимому сообщения аутентифицирует какую-либо информацию (см. параграф 9.2), рассчитывается отпечаток (подпись) для содержимого, а затем этот отпечаток и дополнительные данные аутентифицируются с использованием ключа аутентификации и результат служит значением MAC.

9.1. Тип AuthenticatedData

Ниже приведен идентификатор объекта для содержимого типа authenticated-data.

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

Тип authenticated-data должен иметь форму ASN.1 AuthenticatedData.

```
AuthenticatedData ::= SEQUENCE {
  version CMSVersion,
  originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos RecipientInfos,
  macAlgorithm MessageAuthenticationCodeAlgorithm,
  digestAlgorithm [1] DigestAlgorithmIdentifier OPTIONAL,
  encapContentInfo EncapsulatedContentInfo,
  authAttrs [2] IMPLICIT AuthAttributes OPTIONAL,
  mac MessageAuthenticationCode,
  unauthAttrs [3] IMPLICIT UnauthAttributes OPTIONAL }

AuthAttributes ::= SET SIZE (1..MAX) OF Attribute

UnauthAttributes ::= SET SIZE (1..MAX) OF Attribute

MessageAuthenticationCode ::= OCTET STRING
```

Поля типа AuthenticatedData описаны ниже.

version — номер версии синтаксиса. Значение version **должно** устанавливаться по приведенным ниже правилам

```
IF (присутствует originatorInfo) AND
   ((присутствуют любые сертификаты типа other) OR
   (присутствуют любые crls типа other))
THEN version = 3
ELSE
   IF ((присутствует originatorInfo) AND
      (присутствуют любые сертификаты атрибутов версии 2))
   THEN version = 1
   ELSE version = 0
```

originatorInfo — необязательная информация об инициаторе. Параметр указывается лишь в тех случаях, когда этого требует механизм управления ключами. Поле **может** включать сертификаты, сертификаты атрибутов и CRL, как указано в параграфе 6.1.

recipientInfos — набор информации для получателей, как определено в параграфе 6.1. Набор **должен** включать не менее 1 элемента.

macAlgorithm — идентификатор, указывающий алгоритм MAC и все связанные с ним параметры, использованные инициатором. Размещение поля macAlgorithm облегчает обработку получателем за один проход.

digestAlgorithm указывает алгоритм цифровой подписи для сообщения и все связанные с ним параметры, которые были использованы для расчета отпечатка инкапсулированного содержимого, если присутствуют аутентифицированные атрибуты. Процесс создания цифровой подписи (отпечатка) сообщения описан в параграфе 9.2. Размещение поля digestAlgorithm облегчает получателю обработку за один проход. При наличии поля digestAlgorithm **должно** присутствовать и поле authAttrs.

encapContentInfo — аутентифицируемое содержимое, как определено в параграфе 5.2.

authAttrs — набор аутентифицируемых атрибутов. Структура authAttrs является необязательной, но она **должна** присутствовать, если тип содержимого для значения EncapsulatedContentInfo, которое аутентифицируется, отличается от id-data. При наличии поля authAttrs **должно** присутствовать также поле digestAlgorithm. Для структуры AuthAttributes **должно** использоваться представление DER, даже если остальная часть структуры задана в представлении BER. Полезные типы атрибутов определены в разделе 11. При наличии поля authAttrs оно **должно** содержать по крайней мере два перечисленных ниже атрибута:

content-type использует в качестве своего значения тип аутентифицируемого значения EncapsulatedContentInfo. Атрибут content-type определен в параграфе 11.1.

message-digest использует в качестве своего значения цифровую подпись содержимого. Атрибут message-digest определен в параграфе 11.2.

mac — код аутентификации сообщения.

unauthAttrs — необязательный набор атрибутов, которые не аутентифицируются. В настоящее время не определены неаутентифицированные атрибуты, а другие полезные атрибуты определены в разделе 11.

9.2. Генерация MAC

В процессе расчета MAC вычисляется код аутентификации сообщения (MAC) для аутентифицируемого содержимого или отпечатка аутентифицируемого содержимого вместе с аутентифицируемыми атрибутами инициатора.

Если поле authAttrs отсутствует, входными данными для процесса расчета MAC служит строка октетов encapContentInfo eContent. В расчете MAC используются только октеты строки eContent без учета октетов тега и размера. Это позволяет рассчитывать значение MAC для строк, размер которых не известен заранее процессу MAC.

При наличии поля authAttrs **должны** учитываться атрибуты content-type (параграф 11.1) и message-digest (параграф 11.2), а входной информацией для процесса расчета MAC служит DER-представление authAttrs. Для расчета цифровой подписи сообщения выполняется отдельное кодирование поля authAttrs. Тег IMPLICIT [2] в поле authAttrs не используется для представления DER, взамен его применяется тег EXPLICIT SET OF. Т. е. в расчет цифровой подписи вместо DER-представления тега IMPLICIT [2] включается представление тега SET OF вместе с октетами размера и содержимого authAttrs.

Процесс расчета цифровой подписи вычисляет отпечаток аутентифицируемого сообщения. Исходными данными для процесса является «значение» инкапсулированного содержимого, которое будет аутентифицироваться. А именно, входными данными служит строка октетов encapContentInfo eContent, к которой применяется процесс аутентификации. Алгоритм расчета цифровой подписи учитывает только значение encapContentInfo eContent без октетов тега и размера. Это позволяет рассчитать цифровую подпись для содержимого, размер которого не известен заранее. Хотя октеты тега и размера encapContentInfo eContent не включаются в расчет цифровой подписи, они защищаются другими средствами. Октеты размера защищаются самой природой алгоритма цифровой подписи, поскольку найти два варианта содержимого произвольного размера, которые имели бы одинаковые отпечатки, не представляется возможным.

Входные данные процесса расчета MAC включают определенные выше входные данные MAC и ключ аутентификации, передаваемый в структуре recipientInfo. Детали расчета MAC зависят от используемого алгоритма MAC (например, HMAC). Идентификатор объекта вместе со всеми параметрами, определяющими использованный инициатором алгоритм MAC, передаются в поле macAlgorithm. Значение MAC, инициатором инициатором, передается в форме строки октетов (OCTET STRING) в поле mac.

9.3. Проверка MAC

Входом для процесса проверки MAC служат входные данные (определяются на основе наличия или отсутствия поля authAttrs, как указано в параграфе 9.2) и ключ аутентификации, передаваемый в recipientInfo. Детали процесса проверки MAC зависят от используемого алгоритма MAC.

Получателю **недопустимо** опираться на любые значения MAC или цифровой подписи, рассчитанные инициатором. Содержимое аутентифицируется в соответствии с описанием параграфа 9.2. Если инициатор включил аутентифицируемые атрибуты, содержимое authAttrs аутентифицируется в соответствии с параграфом 9.2. Для того, чтобы аутентификация считалась пройденной, значение MAC, рассчитанное получателем, **должно** совпасть со значением поля mac. Аналогично для успешной аутентификации при наличии поля authAttrs значение цифровой подписи, рассчитанное получателем, **должно** совпадать со значением атрибута message-digest в authAttrs.

Если AuthenticatedData включает authAttrs, значение атрибута content-type **должно** соответствовать значению AuthenticatedData encapContentInfo eContentType.

10. Полезные типы

Этот раздел состоит из двух частей — в первой определены идентификаторы алгоритмов, а во второй другие полезные типы.

10.1. Типы идентификаторов алгоритма

Все идентификаторы алгоритмов используют один тип:

AlgorithmIdentifier — определение заимствовано из X.509 [X.509-88].

Для каждого типа алгоритма имеется множество вариантов.

10.1.1. DigestAlgorithmIdentifier

Тип DigestAlgorithmIdentifier указывает алгоритм message-digest, примерами могут служить алгоритмы SHA-1, MD2, MD5. Алгоритм message-digest отображает строку октетов (содержимое) в другую строку октетов (отпечаток).

```
DigestAlgorithmIdentifier ::= AlgorithmIdentifier
```

10.1.2. SignatureAlgorithmIdentifier

Тип SignatureAlgorithmIdentifier указывает алгоритм подписи и может также указывать алгоритм отпечатка[1] (message digest). Примерами могут служить RSA, DSA, DSA с SHA-1, ECDSA и ECDSA с SHA-256. Алгоритм подписи поддерживает операции создания и проверки подписи. При генерации подписи используется отпечаток сообщения и секретный ключ подписывающего. При проверке подписи используется отпечаток сообщения и открытый ключ подписавшего для проверки пригодности полученной подписи. Выполняемая операция определяется контекстом.

```
SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
```

10.1.3. KeyEncryptionAlgorithmIdentifier

Тип KeyEncryptionAlgorithmIdentifier указывает алгоритм шифрования ключа используемого для шифрования содержимого. Операция шифрования отображает строку октетов (ключ) на другую строку октетов (зашифрованный

[1] В оригинале ошибочно указан только алгоритм подписи. См. https://www.rfc-editor.org/errata_search.php?eid=1756. *Прим. перев.*

ключ) под управлением ключа шифрования ключа. Операция расшифровки обратна операции шифрования. Выбор операции определяется контекстом.

Детали шифрования и расшифровки зависят от используемого алгоритма управления ключами. Поддерживаются доставка (key transport) и согласование (key agreement) ключей, заранее известные симметричные ключи и ключи, создаваемые из паролей.

```
KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
```

10.1.4. ContentEncryptionAlgorithmIdentifier

Тип ContentEncryptionAlgorithmIdentifier указывает алгоритм шифрования содержимого (примерами могут служить Triple-DES и RC2). Алгоритм шифрования содержимого поддерживает операции шифрования и расшифровки. Операция шифрования отображает строку октетов (открытый текст) на другую строку октетов (шифрованный текст) с использованием ключа шифрования содержимого. Операция расшифровки выполняет обратное преобразование. Выбор операции определяется контекстом.

```
ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
```

10.1.5. MessageAuthenticationCodeAlgorithm

Тип MessageAuthenticationCodeAlgorithm указывает алгоритм для кода аутентификации сообщения (MAC) — примерами могут служить DES-MAC и HMAC-SHA-1. Алгоритм MAC поддерживает операции создания и проверки кода, в которых используется один и тот же симметричный ключ. Выбор операции определяется контекстом.

```
MessageAuthenticationCodeAlgorithm ::= AlgorithmIdentifier
```

10.1.6. KeyDerivationAlgorithmIdentifier

Тип KeyDerivationAlgorithmIdentifier определен в RFC 3211 [PWRI] и здесь это определение повторено для полноты.

Алгоритм генерации производных ключей преобразует пароль или общий секрет в ключ шифрования ключей.

```
KeyDerivationAlgorithmIdentifier ::= AlgorithmIdentifier
```

10.2. Другие полезные типы

В этом параграфе определены типы, используемые в других разделах данного документа. Типы никак не упорядочены.

10.2.1. RevocationInfoChoices

Тип RevocationInfoChoices дает набор вариантов информации о статусе отзыва. Предполагается, что такой набор содержит информацию, достаточную для идентификации связанных с набором сертификатов и сертификатов атрибутов, которые отозваны. Однако в наборе **может** содержаться избыточная информация о статусе отзыва, а также набор **может** не включать всей требуемой информации. Списки отзыва сертификатов X.509 (CRL) [X.509-97] являются основным источником информации о статусе отзыва, но могут поддерживаться иные форматы такой информации. Обеспечивается вариант OtherRevocationInfoFormat для поддержки любых других форматов информации об отзыве без дополнительного изменения CMS. Например, с помощью OtherRevocationInfoFormat могут поддерживаться отзывы протокола OCSP[2] [OCSP].

CertificateList может содержать список CRL, ARL[3], Delta CRL или Attribute CRL. Все эти списки используют общий синтаксис.

Тип CertificateList указывает список отзыва сертификатов (CRL). Эти списки определены в X.509 [X.509-97] и заданы для использования в Internet документом RFC 3280 [PROFILE].

Ниже приведено определение CertificateList из X.509.

```
RevocationInfoChoices ::= SET OF RevocationInfoChoice

RevocationInfoChoice ::= CHOICE {
  crl CertificateList,
  other [1] IMPLICIT OtherRevocationInfoFormat }

OtherRevocationInfoFormat ::= SEQUENCE {
  otherRevInfoFormat OBJECT IDENTIFIER,
  otherRevInfo ANY DEFINED BY otherRevInfoFormat }
```

10.2.2. CertificateChoices

Тип CertificateChoices задает расширенный сертификат PKCS #6 [PKCS#6], сертификат X.509, сертификат атрибута X.509 версии 1 (ACv1) [X.509-97], сертификат атрибута X.509 версии 2 (ACv2) [X.509-00] или сертификат иного формата. Расширенные сертификаты PKCS #6 считаются устаревшими и поддерживаются лишь для совместимости со старыми версиями, поэтому применять их **не следует**. Сертификаты ACv1 также являются устаревшими, включены лишь для совместимости со старыми версиями и применять их **не следует**. Профиль Internet для сертификатов X.509 задан в документе Internet X.509 Public Key Infrastructure: Certificate and CRL Profile [PROFILE], профиль для сертификатов ACv2 — в документе An Internet Attribute Certificate Profile for Authorization [ACPROFILE]. Вариант OtherCertificateFormat служит для поддержки других форматов сертификатов без изменения CMS.

Определение Certificate заимствовано из X.509.

Определения AttributeCertificate заимствованы из X.509-1997 и X.509-2000 — определение из X.509-1997 обозначено AttributeCertificateV1 (см. параграф 12.2), а определение из X.509-2000 - AttributeCertificateV2.

```
CertificateChoices ::= CHOICE {
 certificate Certificate,
```

[2]Online Certificate Status Protocol — протокол интерактивной информации о статусе сертификата.
[3]Authority Revocation List — список отзыва УЦ.

```
    extendedCertificate [0] IMPLICIT ExtendedCertificate, -- отменено
    v1AttrCert [1] IMPLICIT AttributeCertificateV1,       -- отменено
    v2AttrCert [2] IMPLICIT AttributeCertificateV2,
    other [3] IMPLICIT OtherCertificateFormat }

OtherCertificateFormat ::= SEQUENCE {
    otherCertFormat OBJECT IDENTIFIER,
    otherCert ANY DEFINED BY otherCertFormat }
```

10.2.3. CertificateSet

Тип CertificateSet представляет набор сертификатов, достаточный для обеспечения «пути сертификации» (certification path) от «корня» или «удостоверяющего центра верхнего уровня» до всех сертификатов отправителя, с которыми связан данный набор. Однако набор **может** включать избыточные сертификаты, а также **может** содержать неполный их комплект.

Точное значение термина «путь сертификации» выходит за рамки данной спецификации. Однако [PROFILE] включает определения для сертификатов X.509. Некоторые приложения могут ограничивать сверху размер пути сертификации, а другие могут предъявлять те или иные требования к связям между субъектами и эмитентами в пути сертификации.

```
CertificateSet ::= SET OF CertificateChoices
```

10.2.4. IssuerAndSerialNumber

Тип IssuerAndSerialNumber указывает сертификат и, таким образом, субъекта и его открытый ключ по отличительному имени (distinguished name) эмитента и порядковому номеру сертификата.

Определение поля Name заимствовано из X.501 [X.501-88], а поля CertificateSerialNumber — из X.509 [X.509-97].

```
IssuerAndSerialNumber ::= SEQUENCE {
  issuer Name,
  serialNumber CertificateSerialNumber }

CertificateSerialNumber ::= INTEGER
```

10.2.5. CMSVersion

Тип CMSVersion указывает номер версии синтаксиса для обеспечения совместимости с будущими версиями этой спецификации.

```
CMSVersion ::= INTEGER { v0(0), v1(1), v2(2), v3(3), v4(4), v5(5) }
```

10.2.6. UserKeyingMaterial

Тип UserKeyingMaterial указывает синтаксис для пользовательского ключевого материала (UKM[1]). Некоторым алгоритмам согласования ключей UKM требуется для генерации разных ключей в каждом случае согласования между данной парой сторон. Отправитель предоставляет UKM для использования с конкретным алгоритмом согласования ключей.

```
UserKeyingMaterial ::= OCTET STRING
```

10.2.7. OtherKeyAttribute

Тип OtherKeyAttribute задает синтаксис для включения других атрибутов ключа, которые позволят получателю выбрать ключ, использованный отправителем. Идентификаторы атрибутов объектов должны регистрироваться вместе с синтаксисом самих атрибутов. Использования этого типа следует избегать, поскольку он может приводить к несовместимости.

```
OtherKeyAttribute ::= SEQUENCE {
  keyAttrId OBJECT IDENTIFIER,
  keyAttr ANY DEFINED BY keyAttrId OPTIONAL }
```

11. Полезные атрибуты

В этом разделе определены атрибуты, которые могут использоваться с содержимым типов signed-data, enveloped-data, encrypted-data, authenticated-data. Синтаксис Attribute совместим с X.501 [X.501-88] и RFC 3280 [PROFILE]. Некоторые из описанных в этом разделе атрибутов были определены в PKCS #9 [PKCS#9], а другие — в предыдущей версии этой спецификации [CMS1]. Атрибуты перечислены без какого-либо упорядочения.

Из числа атрибутов, определенных в других документах, следует отметить спецификацию S/MIME версии 3 [MSG] и расширенные средства защиты для S/MIME [ESS], которые включают также рекомендации по размещению этих атрибутов.

11.1. Тип содержимого

Атрибут content-type указывает тип содержимого ContentInfo в signed-data или authenticated-data. Атрибут content-type **должен** присутствовать при наличии подписанных атрибутов в signed-data или аутентифицированных атрибутов в authenticated-data. Значение атрибута content-type **должно** соответствовать значению encapContentInfo eContentType в signed-data или authenticated-data.

Атрибут content-type **должен** быть подписанным (signed) или аутентифицированным (authenticated) атрибутом, для него **недопустимо** быть неподписанным (unsigned), неаутентифицированным (unauthenticated) или незащищенным (unprotected) атрибутом.

Ниже приведен идентификатор объекта для атрибута content-type.

[1]User keying material.

```
id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }
```

Значения атрибута content-type имеют тип ASN.1 ContentType

```
ContentType ::= OBJECT IDENTIFIER
```

Хотя синтаксис определен, как SET OF AttributeValue, атрибут content-type **должен** иметь единственное значение, отсутствие или несколько экземпляров AttributeValue не допускается.

Синтаксис SignedAttributes и AuthAttributes определяет их как SET OF Attributes. В SignedAttributes поля signerInfo **недопустимо** включать множество экземпляров атрибута content-type. Аналогично, в AuthAttributes поля AuthenticatedData **недопустимо** включать множество экземпляров атрибута content-type.

11.2. Отпечаток сообщения

Тип атрибута message-digest задает отпечаток (message digest) строки октетов encapContentInfo eContent, подписанной в signed-data (см. параграф 5.4) или аутентифицированной в authenticated-data (см. параграф 9.2). Для signed-data отпечаток рассчитывается с использованием алгоритма хэширования подписывающей стороны. Для authenticated-data отпечаток рассчитывается с использованием алгоритма хэширования инициатора.

В signed-data тип атрибута message-digest **должен** присутствовать при наличии любых подписанных атрибутов. В authenticated-data тип атрибута message-digest **должен** присутствовать при наличии любых аутентифицированных атрибутов.

Атрибут message-digest **должен** быть подписанным (signed) или аутентифицированным (authenticated), **недопустимо** использование неподписанного (unsigned), неаутентифицированного (unauthenticated) или незащищенного (unprotected) атрибута.

Ниже приведен идентификатор объекта для атрибута message-digest.

```
id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }
```

Атрибут message-digest должен иметь тип ASN.1 MessageDigest

```
MessageDigest ::= OCTET STRING
```

Атрибут message-digest **должен** иметь единственное значение, несмотря на то, что его синтаксис определен, как SET OF AttributeValue. **Недопустимо** отсутствие или наличие множества экземпляров AttributeValue.

Синтаксис SignedAttributes и AuthAttributes определяет их как SET OF Attributes. В SignedAttributes поля signerInfo **недопустимо** включать множество экземпляров атрибута message-digest. Аналогично, в AuthAttributes поля AuthenticatedData **недопустимо** включать множество экземпляров атрибута message-digest.

11.3. Время подписи

Атрибут signing-time указывает время, когда подписавший (предположительно) «поставил свою подпись». Этот атрибут предназначен для использования в signed-data.

Атрибут signing-time **должен** быть подписанным (signed) или аутентифицированным (authenticated), **недопустимо** использование неподписанного (unsigned), неаутентифицированного (unauthenticated) или незащищенного (unprotected) атрибута.

Ниже приведен идентификатор объекта для атрибута signing-time.

```
id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }
```

Значения атрибыта signing-time имеют тип ASN.1 SigningTime.

```
SigningTime ::= Time

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

Примечание. Определение Time соответствует версии X.509 от 1997 года [X.509-97].

Даты между 1 января 1950 г. и 31 декабря 2049 г. (включительно) **должны** представляться в формате UTCTime. Даты до 1950 г. и после 2049 г. **должны** представляться в формате GeneralizedTime.

Значения UTCTime **должны** указываться по времени Coordinated Universal Time (ранее, гринвичское время - Greenwich Mean Time или GMT, а также время Zulu) и **должны** включать секунды (т. е., использовать формат YYMMDDHHMMSSZ) даже если число секунд равно 0. Полночь **должна** представляться в виде YYMMDD000000Z. Информация о столетии является неявной и **должна** определяться следующим образом:

при YY не меньше 50 значение года **должно** интерпретироваться, как 19YY;

при YY < 50 значение года **должно** интерпретироваться, как 20YY.

Значения GeneralizedTime **должны** представляться во времени Coordinated Universal Time и **должны** включать секунды (т. е., использовать формат YYYYMMDDHHMMSSZ)даже если число секунд равно 0. В значения GeneralizedTime **недопустимо** включать доли секунд.

Атрибут signing-time **должен** иметь единственное значение, несмотря на синтаксис определения SET OF AttributeValue. **Недопустимо** отсутствие или множество экземпляров AttributeValue.

Синтаксис SignedAttributes и AuthAttributes определяет их как SET OF Attributes. В SignedAttributes поля signerInfo **недопустимо** включать множество экземпляров атрибута signing-time. Аналогично, в AuthAttributes поля AuthenticatedData **недопустимо** включать множество экземпляров атрибута signing-time.

Требования к корректности времени подписания не задаются и решение вопроса о доверии к этому времени остается за получателем. Однако неявно предполагается, что некоторым из подписывающих (например, серверам временных меток) можно доверять.

11.4. Заверяющая подпись

Тип атрибута countersignature указывает одну или множество подписей для строки октетов содержимого signature в значении SignerInfo для signed-data. Т. е., отпечаток сообщения рассчитывается для октетов, представляющих значение OCTET STRING, без октетов тега и размера. Таким образом, тип атрибута countersignature «заверяет» (еще раз подписывает) другую подпись.

Атрибут countersignature **должен** быть неподписанным (unsigned), **недопустимо** использование подписанных (signed), аутентифицированных (authenticated), неаутентифицированных (unauthenticated) и незащищенных (unprotected) атрибутов.

Ниже приведен идентификатор объекта для атрибута countersignature.

```
id-countersignature OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 6 }
```

Значения атрибута countersignature имеют тип ASN.1 Countersignature

```
Countersignature ::= SignerInfo
```

Значения countersignature имеют такой же смысл, как значения SignerInfo для обычных подписей, за исключением отмеченных ниже различий.

1. В поле signedAttributes **недопустимо** включать атрибут content-type — для заверяющих подписей нет типа содержимого.
2. Поле signedAttributes **должно** включать атрибут message-digest, если в нем есть какие-либо другие атрибуты.
3. Входными данными для процесса создания отпечатка сообщения являются октеты содержимого DER-представления поля signatureValue в значении SignerInfo, с которым связан атрибут.

Атрибут countersignature может иметь множество значений. Синтаксис определен, как SET OF AttributeValue и **должен** присутствовать хотя бы один экземпляр AttributeValue.

Синтаксис UnsignedAttributes определен, как SET OF Attributes. UnsignedAttributes в signerInfo может включать множество экземпляров атрибута countersignature.

Поскольку countersignature имеет тип SignerInfo, атрибут сам по себе может включать атрибут countersignature. Это позволяет создавать цепочки заверяющих подписей произвольной длины.

12. Модули ASN.1

В параграфе 12.1 приведен модуль ASN.1 для CMS, а в параграфе 12.2 — для сертификата атрибутов версии 1.

12.1. Модуль CMS ASN.1

```
CryptographicMessageSyntax2004
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- Экспортируется все
-- Типы и значения, определенные в этом модуле, экспортируются для использования
-- в других модулях ASN.1. Их могут применять для своих целей другие приложения.

IMPORTS

  -- Импорт из RFC 3280 [PROFILE], Приложение A.1
       AlgorithmIdentifier, Certificate, CertificateList,
       CertificateSerialNumber, Name
          FROM PKIX1Explicit88
               { iso(1) identified-organization(3) dod(6)
                 internet(1) security(5) mechanisms(5) pkix(7)
                 mod(0) pkix1-explicit(18) }

  -- Импорт из RFC 3281 [ACPROFILE], Приложение B
       AttributeCertificate
          FROM PKIXAttributeCertificate
               { iso(1) identified-organization(3) dod(6)
                 internet(1) security(5) mechanisms(5) pkix(7)
                 mod(0) attribute-cert(12) }

  -- Импорт из Приложения B к данному документу
       AttributeCertificateV1
          FROM AttributeCertificateVersion1
               { iso(1) member-body(2) us(840) rsadsi(113549)
                 pkcs(1) pkcs-9(9) smime(16) modules(0)
                 v1AttrCert(15) } ;
```

```
-- Синтаксис криптографического сообщения

ContentInfo ::= SEQUENCE {
  contentType ContentType,
  content [0] EXPLICIT ANY DEFINED BY contentType }

ContentType ::= OBJECT IDENTIFIER

SignedData ::= SEQUENCE {
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,
  encapContentInfo EncapsulatedContentInfo,
  certificates [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos SignerInfos }

DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

SignerInfos ::= SET OF SignerInfo

EncapsulatedContentInfo ::= SEQUENCE {
  eContentType ContentType,
  eContent [0] EXPLICIT OCTET STRING OPTIONAL }

SignerInfo ::= SEQUENCE {
  version CMSVersion,
  sid SignerIdentifier,
  digestAlgorithm DigestAlgorithmIdentifier,
  signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature SignatureValue,
  unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier [0] SubjectKeyIdentifier }

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
  attrType OBJECT IDENTIFIER,
  attrValues SET OF AttributeValue }

AttributeValue ::= ANY

SignatureValue ::= OCTET STRING

EnvelopedData ::= SEQUENCE {
  version CMSVersion,
  originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos RecipientInfos,
  encryptedContentInfo EncryptedContentInfo,
  unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }

OriginatorInfo ::= SEQUENCE {
  certs [0] IMPLICIT CertificateSet OPTIONAL,
  crls [1] IMPLICIT RevocationInfoChoices OPTIONAL }

RecipientInfos ::= SET SIZE (1..MAX) OF RecipientInfo

EncryptedContentInfo ::= SEQUENCE {
  contentType ContentType,
  contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
  encryptedContent [0] IMPLICIT EncryptedContent OPTIONAL }

EncryptedContent ::= OCTET STRING

UnprotectedAttributes ::= SET SIZE (1..MAX) OF Attribute

RecipientInfo ::= CHOICE {
  ktri KeyTransRecipientInfo,
  kari [1] KeyAgreeRecipientInfo,
  kekri [2] KEKRecipientInfo,
```

```
    pwri [3] PasswordRecipientInfo,
    ori [4] OtherRecipientInfo }

  EncryptedKey ::= OCTET STRING

  KeyTransRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- всегда устанавливается значение 0 или 2
    rid RecipientIdentifier,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }

  RecipientIdentifier ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    subjectKeyIdentifier [0] SubjectKeyIdentifier }

  KeyAgreeRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- всегда устанавливается значение 3
    originator [0] EXPLICIT OriginatorIdentifierOrKey,
    ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    recipientEncryptedKeys RecipientEncryptedKeys }

  OriginatorIdentifierOrKey ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    subjectKeyIdentifier [0] SubjectKeyIdentifier,
    originatorKey [1] OriginatorPublicKey }

  OriginatorPublicKey ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    publicKey BIT STRING }

  RecipientEncryptedKeys ::= SEQUENCE OF RecipientEncryptedKey

  RecipientEncryptedKey ::= SEQUENCE {
    rid KeyAgreeRecipientIdentifier,
    encryptedKey EncryptedKey }

  KeyAgreeRecipientIdentifier ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    rKeyId [0] IMPLICIT RecipientKeyIdentifier }

  RecipientKeyIdentifier ::= SEQUENCE {
    subjectKeyIdentifier SubjectKeyIdentifier,
    date GeneralizedTime OPTIONAL,
    other OtherKeyAttribute OPTIONAL }

  SubjectKeyIdentifier ::= OCTET STRING

  KEKRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- всегда устанавливается значение 4
    kekid KEKIdentifier,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }

  KEKIdentifier ::= SEQUENCE {
    keyIdentifier OCTET STRING,
    date GeneralizedTime OPTIONAL,
    other OtherKeyAttribute OPTIONAL }

  PasswordRecipientInfo ::= SEQUENCE {
    version CMSVersion,   -- всегда устанавливается значение 0
    keyDerivationAlgorithm [0] KeyDerivationAlgorithmIdentifier OPTIONAL,
    keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
    encryptedKey EncryptedKey }

  OtherRecipientInfo ::= SEQUENCE {
    oriType OBJECT IDENTIFIER,
    oriValue ANY DEFINED BY oriType }

  DigestedData ::= SEQUENCE {
    version CMSVersion,
    digestAlgorithm DigestAlgorithmIdentifier,
    encapContentInfo EncapsulatedContentInfo,
    digest Digest }

  Digest ::= OCTET STRING
```

```
EncryptedData ::= SEQUENCE {
  version CMSVersion,
  encryptedContentInfo EncryptedContentInfo,
  unprotectedAttrs [1] IMPLICIT UnprotectedAttributes OPTIONAL }

AuthenticatedData ::= SEQUENCE {
  version CMSVersion,
  originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos RecipientInfos,
  macAlgorithm MessageAuthenticationCodeAlgorithm,
  digestAlgorithm [1] DigestAlgorithmIdentifier OPTIONAL,
  encapContentInfo EncapsulatedContentInfo,
  authAttrs [2] IMPLICIT AuthAttributes OPTIONAL,
  mac MessageAuthenticationCode,
  unauthAttrs [3] IMPLICIT UnauthAttributes OPTIONAL }

AuthAttributes ::= SET SIZE (1..MAX) OF Attribute

UnauthAttributes ::= SET SIZE (1..MAX) OF Attribute

MessageAuthenticationCode ::= OCTET STRING

DigestAlgorithmIdentifier ::= AlgorithmIdentifier

SignatureAlgorithmIdentifier ::= AlgorithmIdentifier

KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier

MessageAuthenticationCodeAlgorithm ::= AlgorithmIdentifier

KeyDerivationAlgorithmIdentifier ::= AlgorithmIdentifier

RevocationInfoChoices ::= SET OF RevocationInfoChoice

RevocationInfoChoice ::= CHOICE {
  crl CertificateList,
  other [1] IMPLICIT OtherRevocationInfoFormat }

OtherRevocationInfoFormat ::= SEQUENCE {
  otherRevInfoFormat OBJECT IDENTIFIER,
  otherRevInfo ANY DEFINED BY otherRevInfoFormat }

CertificateChoices ::= CHOICE {
  certificate Certificate,
  extendedCertificate [0] IMPLICIT ExtendedCertificate,  -- Отменено
  v1AttrCert [1] IMPLICIT AttributeCertificateV1,        -- Отменено
  v2AttrCert [2] IMPLICIT AttributeCertificateV2,
  other [3] IMPLICIT OtherCertificateFormat }

AttributeCertificateV2 ::= AttributeCertificate

OtherCertificateFormat ::= SEQUENCE {
  otherCertFormat OBJECT IDENTIFIER,
  otherCert ANY DEFINED BY otherCertFormat }

CertificateSet ::= SET OF CertificateChoices

IssuerAndSerialNumber ::= SEQUENCE {
  issuer Name,
  serialNumber CertificateSerialNumber }

CMSVersion ::= INTEGER  { v0(0), v1(1), v2(2), v3(3), v4(4), v5(5) }

UserKeyingMaterial ::= OCTET STRING

OtherKeyAttribute ::= SEQUENCE {
  keyAttrId OBJECT IDENTIFIER,
  keyAttr ANY DEFINED BY keyAttrId OPTIONAL }

-- Идентификаторы типа содержимого

id-ct-contentInfo OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1) 6 }
```

```
id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }

id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }

id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }

id-digestedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 5 }

id-encryptedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs7(7) 6 }

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

-- Атрибуты CMS

MessageDigest ::= OCTET STRING

SigningTime  ::= Time

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

Countersignature ::= SignerInfo

-- Идентификаторы атрибутов объекта

id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }

id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }

id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }

id-countersignature OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs9(9) 6 }

-- Устаревший расширенный синтаксис из PKCS#6

ExtendedCertificateOrCertificate ::= CHOICE {
  certificate Certificate,
  extendedCertificate [0] IMPLICIT ExtendedCertificate }

ExtendedCertificate ::= SEQUENCE {
  extendedCertificateInfo ExtendedCertificateInfo,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature Signature }

ExtendedCertificateInfo ::= SEQUENCE {
  version CMSVersion,
  certificate Certificate,
  attributes UnauthAttributes }

Signature ::= BIT STRING

END -- для CryptographicMessageSyntax2004
```

12.2. Модуль ASN.1 для сертификата атрибута версии 1

```
AttributeCertificateVersion1
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) v1AttrCert(15) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

-- Экспортируется все

IMPORTS
```

```
   -- Импорт из RFC 3280 [PROFILE], Приложение A.1
         AlgorithmIdentifier, Attribute, CertificateSerialNumber,
         Extensions, UniqueIdentifier
            FROM PKIX1Explicit88
                 { iso(1) identified-organization(3) dod(6)
                   internet(1) security(5) mechanisms(5) pkix(7)
                   mod(0) pkix1-explicit(18) }

   -- Импорт из RFC 3280 [PROFILE], Приложение A.2
         GeneralNames
            FROM PKIX1Implicit88
                 { iso(1) identified-organization(3) dod(6)
                   internet(1) security(5) mechanisms(5) pkix(7)
                   mod(0) pkix1-implicit(19) }

   -- Импорт из RFC 3281 [ACPROFILE], Приложение B
         AttCertValidityPeriod, IssuerSerial
            FROM PKIXAttributeCertificate
                 { iso(1) identified-organization(3) dod(6)
                   internet(1) security(5) mechanisms(5) pkix(7)
                   mod(0) attribute-cert(12) } ;

-- Определения взяты из X.509-1997 [X.509-97], но используются другие имена
-- в целях предотвращения конфликтов.

AttributeCertificateV1 ::= SEQUENCE {
  acInfo AttributeCertificateInfoV1,
  signatureAlgorithm AlgorithmIdentifier,
  signature BIT STRING }

AttributeCertificateInfoV1 ::= SEQUENCE {
  version AttCertVersionV1 DEFAULT v1,
  subject CHOICE {
    baseCertificateID [0] IssuerSerial,
      -- связано с сертификатом открытого ключа
    subjectName [1] GeneralNames },
      -- связано с именем
  issuer GeneralNames,
  signature AlgorithmIdentifier,
  serialNumber CertificateSerialNumber,
  attCertValidityPeriod AttCertValidityPeriod,
  attributes SEQUENCE OF Attribute,
  issuerUniqueID UniqueIdentifier OPTIONAL,
  extensions Extensions OPTIONAL }

AttCertVersionV1 ::= INTEGER { v1(0) }

END -- для AttributeCertificateVersion1
```

13. Литература

13.1. Нормативные документы

[ACPROFILE] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.

[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.

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

[X.208-88] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.

[X.209-88] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

[X.501-88] CCITT. Recommendation X.501: The Directory — Models. 1988.

[X.509-88] CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

[X.509-97] ITU-T. Recommendation X.509: The Directory - Authentication Framework. 1997.

[X.509-00] ITU-T. Recommendation X.509: The Directory - Authentication Framework. 2000.

13.2. Дополнительная литература

[CMS1] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[CMS2] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

[CMSALG] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[MSAC] Microsoft Development Network (MSDN) Library, "Authenticode", April 2004 Release.

[MSG] Ramsdell, B., "S/MIME Version 3.1 Message Specification", RFC 3851, July 2004.

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[OLDMSG] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.

[PKCS#6] RSA Laboratories. PKCS #6: Extended-Certificate Syntax Standard, Version 1.5. November 1993.

[PKCS#7] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[PKCS#9] RSA Laboratories. PKCS #9: Selected Attribute Types, Version 1.1. November 1993.

[PWRI] Gutmann, P., "Password-based Encryption for CMS", RFC 3211, December 2001.

[RANDOM] Eastlake 3rd, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750[1], December 1994.

14. Вопросы безопасности

Синтаксис криптографических сообщений обеспечивает способ цифровой подписи, создания отпечатков, шифрования и аутентификации данных.

Реализации должны обеспечивать защиту секретных ключей подписывающего. Утечка секретного ключа позволит злоумышленникам представиться другим лицом.

Реализации должны защищать секретный ключ управления ключами, ключ шифрования ключей, а также ключ шифрования содержимого. Компрометация секретного ключа управления ключами или шифрования ключей может приводить к раскрытию всего содержимого, защищаемого с помощью этого ключа. Аналогично, компрометация ключа шифрования содержимого может приводить к раскрытию соответствующей зашифрованной информации.

Реализации должны защищать секретный ключ управления ключами и ключ аутентификации сообщений. Компрометация секретного ключа управления ключами может приводить к подмене аутентифицированных данных неаутентифицированными. Аналогично, компрометация ключа аутентификации сообщений может приводить к необнаруживаемому изменению аутентифицированного содержимого.

Метод управления ключами, используемый для распространения ключей аутентификации сообщений, сам по себе должен обеспечивать аутентификацию источников, поскольку в противном случае будут доставляться целостные сообщения из неизвестных источников. Алгоритмы RSA [PKCS#1, NEWPKCS#1] и Ephemeral-Static Diffie-Hellman [DH-X9.42] не обеспечивают требуемой аутентификации источников. Алгоритм Static-Static Diffie-Hellman [DH-X9.42] обеспечивает требуемую аутентификацию, когда ключи инициатора и получателя оба привязаны к приемлемой идентификации в сертификатах X.509.

При использовании общего ключа аутентификации более, чем двумя сторонами, аутентификация источника не обеспечивается. Любая из сторон, которой известен общий ключ аутентификации, может рассчитать корректный код MAC, следовательно, источником может служить любая из таких сторон.

Реализации должны генерировать случайные значения ключей шифрования содержимого, ключей аутентификации сообщений, векторов инициализации (IV) и заполнения. Генерация ключевых пар «открытый-секретный» также основана на случайных значениях. Использование некачественных генераторов псевдослучайных чисел (PRNG[2]) при создании криптографических ключей может приводить к снижению уровня или полной утрате защиты. Атакующему может оказаться гораздо проще воспроизвести среду PRNG, в которой создавались ключи, перебрав сравнительно небольшой набор вариантов, нежели использовать средства подбора (brute force) во всем пространстве возможных ключей. Генерация качественных случайных чисел достаточно сложна. В RFC 1750 [RANDOM] представлены важные рекомендации по решению таких задач.

При использовании алгоритмов согласования ключей или заранее известных ключей шифрования ключей, такие ключи служат для шифрования ключей, которые, в свою очередь, применяются для шифрования содержимого. Если для шифрования ключей и содержимого применяются разные алгоритмы, эффективный уровень защиты определяется наиболее слабым из этих двух алгоритмов. Если, например, содержимое шифруется с помощью алгоритма Triple-DES с использованием 168-битового ключа, а для ключа шифрования ключей применяется алгоритм RC2 с 40-битовым ключом, уровень защиты будет соответствовать 40-битовому ключу. Тривиальный поиск для определения 40-битового ключа RC2 позволит расшифровать ключ Triple-DES, который после этого может использоваться для расшифровки содержимого. Следовательно, разработчики должны гарантировать, что для шифрования ключей используется алгоритм не менее (а возможно и более) сильный, нежели для шифрования содержимого.

Разработчикам следует помнить, что стойкость криптографических алгоритмов со временем ослабевает. Разработка новых методов криптоанализа и повышение производительности компьютеров уменьшают время, потребное для взлома того или иного криптографического алгоритма. Следовательно, реализации криптоалгоритмов должны быть модульными, чтобы обеспечить возможность использования новых алгоритмов. Т. е., реализации должны быть готовы к смене поддерживаемых алгоритмов с течением времени.

Неподписанный атрибут заверяющей подписи включает в себя цифровую подпись, которая рассчитывается для значения подписи содержимого, поэтому процессу заверения подписи не требуется знать исходное содержимое. Такой подход обеспечивает реализациям эффективное преимущество, однако он может позволить также заверение неприемлемой подписи. Следовательно, реализациям, использующим заверяющие подписи, следует проверять исходное значение подписи до ее заверения (это требует обработки исходного содержимого) или заверять подписи лишь в контексте, который обеспечивает корректность заверяемых подписей.

[1]Этот документ заменен RFC 4086. *Прим. перев.*
[2]Pseudo-random number generators.

15. Благодарности

Этот документ включает результаты работы многих профессионалов. Автор отмечает значительный вклад всех членов рабочей группы IETF S/MIME. Отдельные благодарности Rich Ankney, Simon Blake-Wilson, Tim Dean, Steve Dusse, Carl Ellison, Peter Gutmann, Bob Jueneman, Stephen Henson, Paul Hoffman, Scott Hollenbeck, Don Johnson, Burt Kaliski, John Linn, John Pawling, Blake Ramsdell, Francois Rousseau, Jim Schaad, Dave Solo, Paul Timmel и Sean Turner за поддержку с их стороны.

16. Адрес автора

Russell Housley

Vigil Security, LLC

918 Spring Knoll Drive

Herndon, VA 20170

USA

EMail: housley@vigilsec.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

17. Полное заявление авторских прав

Интеллектуальная собственность

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.