ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Р
(проект, первая редакция)

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование алгоритмов ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015, ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в сообщениях формата CMS

Издание официальное

Москва
Стандартинформ
2018

Предисловие

1 РАЗРАБОТАНЫ Открытым акционерным обществом «Информационные технологии и коммуникационные системы» (ОАО «ИнфоТеКС»)

2 ВНЕСЕНЫ Техническим комитетом по стандартизации ТК 26 «Криптографическая защита информации»

3 УТВЕРЖДЕНЫ И ВВЕДЕНЫ В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 20 г. № -ст

4 ВВЕДЕНЫ ВПЕРВЫЕ

Правила применения настоящих рекомендаций установлены в ГОСТ Р 1.0–2012 (раздел 8). Информация об изменениях к настоящим рекомендациям публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящих рекомендаций соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)

Содержание

Введение

Настоящий документ содержит описание форматов кодирования, идентификаторов и форматов параметров при использовании ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015 для криптографической защиты сообщений (сообщений в формате CMS - Cryptographic Message Syntax).

Необходимость разработки новой версии настоящих рекомендаций вызвана введением в действие стандартов ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015, а также необходимостью добавления механизмов контроля целостности содержимого и времени формирования подписей в формат защищенных сообщений.

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Использование алгоритмов ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015, ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 в сообщениях формата CMS

Information technology. Cryptographic data security. Usage of GOST R 34.12-2015, GOST R 34.13-2015, GOST R 34.10-2012 and GOST R 34.11-2012 algorithms in CMS format messages

Дата введения —

1 Область применения

Описанный в настоящих рекомендациях формат предназначен для представления данных, защищенных с помощью криптографических механизмов.

В документе описываются правила использования алгоритмов формирования и проверки подписи в соответствии с ГОСТ Р 34.10-2012, функции хэширования по ГОСТ Р 34.11-2012, функций шифрования в соответствии с ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 для защиты сообщений в формате CMS.

2 Нормативные ссылки

В настоящих рекомендациях использованы нормативные ссылки на следующие документы по стандартизации:

ГОСТ Р 34.10-2012 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.

ГОСТ Р 34.11-2012 Информационная технология. Криптографическая защита информации. Функция хэширования.

Издание официальное

ГОСТ Р 34.12-2015 Информационная технология. Криптографическая защита информации. Блочные шифры.

ГОСТ Р 34.13-2015 Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров.

ГОСТ Р ИСО/МЭК 8824-1 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации.

ГОСТ Р ИСО/МЭК 8825-1 Информационная технология. Правила кодирования АСН.1. Часть 1. Спецификация базовых (BER), канонических (CER) и отличительных (DER) правил кодирования.

ГОСТ Р ИСО/МЭК 9594-2 Информационная технология. Взаимосвязь открытых систем. Справочник: Модели.

ГОСТ Р ИСО/МЭК 9594-8 Информационная технология. Взаимосвязь открытых систем. Справочник. Часть 8. Основы аутентификации.

ГОСТ Р ИСО/МЭК 14888-1:2008 Информационная технология. Методы и средства обеспечения безопасности. Сетевая безопасность информационных технологий.

Р 50.1.113-2016 Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов электронной цифровой подписи и функции хэширования.

Р 50.1.114-2016 Информационная технология. Криптографическая защита информации. Параметры эллиптических кривых для криптографических алгоритмов и протоколов.

П р и м е ч а н и е — При пользовании настоящими рекомендациями целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящих рекомендаций в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

3 Термины, определения и обозначения

3.1 Термины и определения

В настоящих рекомендациях применены следующие термины с соответствующими определениями.

3.1.1 **электронная подпись (ЭП)** - информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.

3.1.2 **ключ электронной подписи** - уникальная последовательность символов, предназначенная для создания электронной подписи.

3.1.3 **ключ проверки электронной подписи** - уникальная последовательность символов, однозначно связанная с ключом электронной подписи и предназначенная для проверки подлинности электронной подписи (далее – проверка электронной подписи).

3.1.4 **сертификат ключа проверки электронной подписи** - электронный документ или документ на бумажном носителе, выданные удостоверяющим центром либо доверенным лицом удостоверяющего центра и подтверждающие принадлежность ключа проверки электронной подписи владельцу сертификата ключа проверки электронной подписи.

3.1.5 **хэш-код** - строка бит, являющаяся выходным результатом хэш-функции (см. ГОСТ Р ИСО/МЭК 14888-1:2008).

3.1.6 **хэш-функция** - функция, отображающая строки бит в строки бит фиксированной длины и удовлетворяющая следующим свойствам (см. ГОСТ Р ИСО/МЭК 14888-1:2008):

- По данному значению функции сложно вычислить исходные данные, отображаемые в это значение;
- Для заданных исходных данных сложно вычислить другие исходные данные, отображаемые в то же значение функции;
- Сложно вычислить какую-либо пару исходных данных, отображаемых в одно и то же значение.

3.2 Обозначения

В настоящих рекомендациях используют следующие обозначения:

V^*	— множество всех двоичных строк конечной длины, включая пустую строку;

V_s	— множество всех двоичных строк длины s, где s — целое неотрицательное число; нумерация подстрок и компонент строки осуществляется справа налево, начиная с нуля;
$\|A\|$	– число компонент (длина) строки $A \in V^*$ (если A – пустая строка, то $\|A\| = 0$);
$A\|\|B$	– конкатенация строк $A, B \in V^*$, т.е. строка из $V_{\|A\|+\|B\|}$, в которой подстрока с большими номерами компонент из $V_{\|A\|}$ совпадает со строкой A, а подстрока с меньшими номерами компонент из $V_{\|B\|}$ совпадает со строкой B;
K_s	– ключ электронной подписи отправителя сообщения;
K_r	–ключ электронной подписи получателя сообщения;
P_s	–ключ проверки электронной подписи отправителя сообщения, представляющий собой точку на эллиптической кривой;
P_r	–ключ проверки электронной подписи получателя сообщения, представляющий собой точку на эллиптической кривой;
UKM	– ключевой материал;
IV	– значение синхропосылки.

4 Общее описание

Данный документ определяет структуру защищенных данных `ContentInfo`. Структура `ContentInfo` инкапсулирует один тип содержимого, данный тип также может быть подвергнут дальнейшей инкапсуляции. В настоящих рекомендациях рассматривается пять типов содержимого:

- Простые данные (`id-data`)
- Подписанные данные (`signed-data`);
- Конверт данных (`enveloped-data`);
- Хэшированные данные (`digested-data`);
- Зашифрованные данные (`encrypted-data`);
- Аутентифицированные данные (`authenticated-data`).

Для каждого из типов содержимого в данном документе рассматриваются варианты использования ГОСТ алгоритмов и механизмов, способы формирования структур данных и параметров и кодирование полученных структур. Для удобства использования описываемых структур принято использовать BER-кодирование (см. ГОСТ Р ИСО/МЭК 8825-1), при котором проверку структуры можно осуществлять за один проход, но для атрибутов структур типа подписанные данные и аутентифицированные данные, в которых может находиться один или несколько нераспознанных получателем атрибутов, требуется использовать DER-кодирование (см. ГОСТ Р ИСО/МЭК 8825-1).

Также в данном документе рассмотрены проблемы использования только одного типа содержимого для формирования сообщения, и даны рекомендации по совместному использованию наборов вложенных типов содержимого для достижения заданных свойств сообщений.

5 Общий синтаксис

Следующий идентификатор определяет тип содержимого:

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

Формат CMS связывает идентификатор типа содержимого с самим содержимым. Тип `ContentInfo` в формате АСН.1 представляется следующим образом:

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

     ContentType ::= OBJECT IDENTIFIER
```

Поля структуры `ContentInfo` имеют следующий смысл:

`contentType` - тип соответствующего содержимого;

`content` - соответствующее содержимое. Тип содержимого может быть однозначно определен по идентификатору в поле `contentType`. В данном документе рассматриваются следующие типы содержимого: «простые данные» (`id-data`), «подписанные данные» (`signed-data`), «конверт данных» (`enveloped-data`), «хэшированные данные» (`digested-data`), «зашифрованные данные» (`encrypted-data`), «аутентифицированные данные» (`authenticated-data`).

6 Содержимое типа «простые данные»

Содержимое типа «простые данные» определяется следующим идентификатором:

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

Тип содержимого «простые данные» предназначен для обозначения произвольной строки байт, например, текстовые файлы ASCII. Такие строки не должны иметь никакой значимой для CMS формата внутренней структуры (хотя они могут иметь собственные внутренние структуры).

7 Содержимое типа «подписанные данные»

Содержимое типа «подписанные данные» представляет собой данные любого типа и любое количество подписей. Любое количество отправителей, осуществляющих подпись, могут подписывать произвольное содержимое независимо.

Примером применения типа «подписанные данные» является использование электронной подписи содержимого в сертификатах и списках аннулированных сертификатов (далее САС).

Процесс построения содержимого типа «подписанные данные» состоит из следующих шагов:

- Для каждого отправителя вычисляется значение хэш-функции от содержимого или от содержимого и подписываемых атрибутов (см. раздел7.4) с использованием алгоритма хэширования, определенного этим отправителем. Если субъекту необходимо подписать кроме содержимого дополнительную информацию, то содержимое хэшируется вместе с этой информацией (см. раздел 7.1), и результат является значением хэш-функции.
- Для каждого отправителя полученное значение хэш-функции подписывается с использованием его ключа ЭП.
- Для каждого отправителя значение подписи и другая специфичная для данного отправителя информация записывается в структуру `SignerInfo` (см. раздел 7.3). На этом шаге также записываются сертификаты и САС как для отправителя, осуществившего подпись, так и для остальных отправителей.

- Значения хэш-функций для всех отправителей и структура `SignerInfo` для всех отправителей записывается вместе с содержимым в структуру `SignedData` (см. раздел 7.47.1).

Получатели, осуществляющие проверку подписи, независимо (самостоятельно) вычисляют значение хэш-функции. Вычисленное значение и ключ проверки ЭП подписавшего отправителя используются для проверки подписи. Ключ проверки ЭП отправителя может быть определен одним из двух способов:

- при помощи уникального имени издателя и серийного номера сертификата, которые однозначно указывают на сертификат с ключом проверки ЭП отправителя;
- при помощи идентификатора ключа субъекта `(Subject Key Identifier)`, который однозначно идентифицирует ключ проверки ЭП отправителя.

Сертификат отправителя, осуществившего подпись, может быть включен в структуру SignedData. Данное включение не является обязательным. Если в сообщении присутствует несколько подписей, то успешная проверка одной подписи, связанная с данным подписавшим, обычно трактуется как успешная проверка подписи отправителя. Тем не менее, есть некоторые приложения, где действуют другие правила проверки подписи отправителя. Приложение, которое использует правило, отличное от правила с одной действительной подписью для каждого подписавшего, должно явно указать эти правила. Детали указания и использования этих правил должны регламентироваться теми приложениями, которые их вводят. В данном документе не рассматриваются. Кроме того, в случае если простое соответствие идентификатора отправителя не является достаточным условием для определения, были ли сгенерированы подписи тем же подписавшим, в спецификации приложения должно быть описано, как определить, какие подписи были получены одним и тем же отправителем. Поддержка различных групп получателей является основной причиной того, что издатели включают более одной подписи. Например, тип содержимого «подписанные данные» может включать в себя подписи, созданные с помощью алгоритма подписи ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012. Это позволяет получателям проверять подписи, полученные с помощью одного или другого алгоритма.

7.1 Тип SignedData

Следующий идентификатор определяет тип содержимого `SignedData`:

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

Содержимое типа «подписанные данные» представляется в виде структуры `SignedData`. Структура `SignedData` в формате АСН.1 представляется следующим образом:

```
SignedData ::= SEQUENCE

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

    DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

    DigestAlgorithmIdentifier := AlgorithmIdentifier

    SignerInfos ::= SET OF SignerInfo
```

Типы `CMSVersion`, `AlgorithmIdentifier`, `CertificateSet`, `RevocationInfoChoices` определены в разделе 14. Тип `EncapsulatedContentInfo` определен в разделе **Error! Reference source not found.**, тип `SignerInfo` определен в разделе 7.3.

Поля структуры `SignedData` имеют следующий смысл:

`version` - версия синтаксиса. Точное значение зависит от полей `CertificateSet`, `eContentType` и `SignerInfo`. Версия синтаксиса должна определяться согласно [1], раздел 5.1.

`digestAlgorithms` - набор идентификаторов алгоритмов хэширования. Набор может состоять из любого числа элементов, в том числе может быть пустым. Каждый элемент определяет алгоритм хэширования со своими параметрами, используемыми одним или несколькими отправителями. Набор идентификаторов предназначен для указания алгоритмов хэширования, используемых всеми получателями, в любом порядке для облегчения проверки подписи за один проход. Подписи, использующие алгоритм хэширования, который не входит в данный набор, могут не проверяться. Процесс вычисления значения хэш-функции описан в разделе 7.4.

`encapContentInfo` - подписываемое содержимое; состоит из идентификатора типа содержимого и самого содержимого. Подробнее структура `EncapsulatedContentInfo` описана в разделе **Error! Reference source not found.**.

`certificates` - набор сертификатов. Предполагается, что множество сертификатов достаточно, чтобы построить цепочки сертификатов от корня или от центра

сертификации верхнего уровня до всех субъектов в `SignerInfo`. Сертификатов может быть больше, чем это необходимо для построения цепочки сертификатов от двух или более центров сертификации верхнего уровня. Сертификатов может быть меньше, чем это необходимо, если получатели имеют альтернативные способы получения необходимых сертификатов (например, из предыдущего набора сертификатов). Сертификат отправителя, подписавшего сообщение, также может быть включен в сообщение.

`crls` - списки аннулированных сертификатов. Предполагается, что CAC достаточно, чтобы проверить корректность сертификата на отзыв, но эта проверка не обязательна.

`signerInfos` - данные о каждом субъекте подписи. Список может состоять из любого числа элементов, в том числе может быть пустым. Подробнее о `SignerInfo` см. раздел 7.3. Поскольку каждый отправитель может использовать различные методы электронной подписи, и будущие спецификации могут обновить синтаксис, все реализации должны корректно обрабатывать неподдерживаемые версии `SignerInfo`. Все реализации должны корректно обрабатывать неподдерживаемые алгоритмы подписи, когда они встречаются.

7.2 Тип EncapsulatedContentInfo

Тип `EncapsulatedContentInfo` в формате ACH.1 представляется следующим образом:

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

     ContentType ::= OBJECT IDENTIFIER
```

Поля структуры `EncapsulatedContentInfo` имеют следующий смысл:

`eContentType` - идентификатор объекта, однозначно определяющий тип содержимого.

`eContent` - содержимое, представленное в виде строки октетов. Содержимое не обязано кодироваться в DER.

Опциональный пропуск параметра `eContent` позволяет создавать так называемые «отделяемые подписи». В случае внешней подписи подписываемое содержимое отсутствует в структуре `EncapsulatedContentInfo`. Если подписываемое содержимое в составе структуры `EncapsulatedContentInfo` не представлено, то тип содержимого, параметр `eContentType`, должен быть указан в соответствии с реальным содержимым.

В вырожденном случае, где нет ни одного субъекта подписи, использование значения `EncapsulatedContentInfo` неуместно. В этом случае тип подписываемого содержимого в `EncapsulatedContentInfo` должен представлять собой «простые данные» (см. 6), а содержимое поля `EncapsulatedContentInfo` должно быть опущено.

7.3 Тип SignerInfo

Информация для каждого подписавшего представляется в виде структуры `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
}

     SubjectKeyIdentifier ::= OCTET STRING

     DigestAlgorithmIdentifier := AlgorithmIdentifier

     SignatureAlgorithmIdentifier := AlgorithmIdentifier

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

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

     SignatureValue ::= OCTET STRING
```

Типы `CMSVersion`, `AlgorithmIdentifier`, `CertificateSet`, `RevocationInfoChoices`, `IssuerAndSerialNumber`, `Attribute` определены в разделе 14.

Поля структуры `SignerInfo` имеют следующий смысл:

`version` - версия синтаксиса; значение данного параметра зависит от варианта указания ключа проверки ЭП отправителя (более подробно см. поле `sid`, определенное ниже). Версия синтаксиса должна определяться согласно [1], раздел 5.3.

`sid` - поле, определяющее сертификат отправителя (и соответственно его ключ проверки ЭП). Ключ проверки ЭП отправителя необходим получателю для проверки подписи. `SignerIdentifier` предоставляет два варианта указания ключа проверки ЭП отправителя. Первый вариант `issuerAndSerialNumber` идентифицирует отправителя по выделенному имени издателя (см. ГОСТ Р ИСО/МЭК 9594-2:2005) и порядковому номеру. Второй вариант `subjectKeyIdentifier` идентифицирует ключ проверки ЭП отправителя. При ссылке на X.509 сертификат идентификатор ключа должен совпадать с расширением `subjectKeyIdentifier` сертификата. При использовании других ссылок на сертификат документы, которые определяют формат сертификата и его использование с CMS, должны включать сведения о соответствии идентификатора ключа и соответствующего поля сертификата. Реализации должны поддерживать оба варианта: `subjectKeyIdentifier` и `issuerAndSerialNumber`. При создании `sid` реализации могут поддерживать одну из форм (либо `subjectKeyIdentifier`, либо `issuerAndSerialNumber`) и всегда использовать ее, или реализации могут смешивать две формы. Тем не менее, `subjectKeyIdentifier` должен использоваться для идентификации ключа проверки ЭП, который не соответствует формату сертификатов X.509.

`digestAlgorithm` - определение алгоритма вычисления хэш-функции и его параметров, используемых отправителем (см. ГОСТ Р 34.11-2012 для отечественных алгоритмов). Значение хэш-функции вычисляется либо от содержимого, либо от содержимого вместе с подписанными атрибутами; этот процесс описан в разделе 7.4. Идентификатор алгоритма хэширования должен присутствовать в числе тех, которые перечислены в поле `digestAlgorithms` структуры `SignedData`. Реализации могут не проверять подписи, использующие алгоритм хэш-функции, который не входит во множество `SignedData.digestAlgorithms`.

`signedAttrs` - набор подписанных атрибутов для подписи. Поле не является обязательным, но оно должно присутствовать, если тип содержимого структуры `EncapsulatedContentInfo` не представляет собой «простые данные».

Поле `signedAttrs` должно быть представлено в DER-кодировке, даже если остальная часть структуры кодируется в BER-кодировке. Если поле присутствует, то оно должно содержать, как минимум два атрибута:

- `content-type` - атрибут, определяющийся значением типа содержимого `EncapsulatedContentInfo` (см. раздел 13.1). Атрибут `content-type` не должен быть использован в не подписываемых атрибутах удостоверяющих подписей (см. раздел 13.3).
- `message-digest` - атрибут, содержащий значение хэш-функции от содержимого (см. раздел 13.2).

`signatureAlgorithm` - определение алгоритма подписи и его параметров, используемых отправителем при подписи.

`signature` - результат вычисления подписи с использованием значения хэш-функции и ключа ЭП отправителя. Подробности алгоритма вычисления подписи зависят от алгоритма (см. раздел 7.7).

`unsignedAttrs` - набор неподписанных атрибутов. Поле не является обязательным.

Поля типа `SignedAttributes и UnsignedAttributes` имеют следующий смысл:

`attrType` - тип атрибута.

`attrValues` - значения атрибутов. Тип каждого значения определяется полем `attrType`. Поле `attrType` может наложить ограничение на количество элементов в наборе.

7.4 Процесс вычисления значения хэш-функции

Процесс вычисления значения хэш-функции состоит из вычисления хэш-функции либо от содержимого, либо от содержимого и от подписываемых атрибутов. В любом случае вначале хэшируется содержимое для подписи, а именно значение поля `SignedData.encapContentInfo.eContent`, представляющее собой строку октетов, к которому применяется процесс вычисления подписи. Хэшируется только содержимое строки октетов без хэширования октетов тега и длины.

Результат вычисления значения хэш-функции зависит от наличия поля `signedAttrs`:

- Если поле отсутствует, то результатом будет являться значение хэш-функции, как описано выше. В этом случае только те октеты, которые составляют значение `SignedData.encapContentInfo.eContent` (например, содержимое файла) являются входом для вычисления значения хэш-функции. Преимущество этого варианта в том, что длина подписываемого содержимого может быть не известна до начала процесса формирования подписи.

- Если поле присутствует, то результатом будет являться значение хэш-функции от значения поля `signedAtrtrs`, представленного в DER-кодировке. Так как поле `signedAttrs` должно содержать атрибуты `content-type` и `message-digest`, то эти значения также должны быть включены в хэшируемое значение. Атрибут `content-type` не должен включаться в неподписываемый атрибут `countersignature`, как это определено в разделе 13.3. Кодирование поля `signedAttrs` выполняется отдельно для вычисления значения хэш-функции. Для DER кодирования в `signedAttrs` вместо тега `IMPLICIT [0]` используется тег `EXPLICIT SET OF`, который должен быть включен в процесс вычисления значения хэш-функции вместе с длиной и строкой октетов поля `SignedAttributes`.

Несмотря на то, что октеты тега и длины `SignedData.encapContentInfo.eContent` не включены в процесс получения значения хэш-функции, они защищены другими средствами. Целостность значения поля длины обуславливается криптографическими свойствами функции хэширования, поскольку алгоритмически сложно найти какие-либо два значения поля длины, которые имеют одинаковые значения хэш-функции.

7.5 Процесс формирования подписи

Для формирования подписи на вход алгоритма подписи необходимы значение хэш-функции и ключ ЭП отправителя. Детали процесса формирования подписи зависят от используемого алгоритма подписи (см. раздел 7.7). Идентификатор объекта, наряду со всеми параметрами, которые задают алгоритм подписи, находится в поле `signatureAlgorithm`.

7.6 Процесс проверки подписи

Для проверки подписи на вход алгоритма проверки подписи необходимы сформированная подпись, значение хэш-функции и ключ проверки ЭП отправителя. Получатель может получить корректный ключ проверки ЭП отправителя любыми средствами, но предпочтительно получение ключа из сертификата из соответствующего поля структуры `SignedData`. Выбор и проверка ключа проверки ЭП отправителя может быть основана на построении и проверке цепочки сертификатов, а также может зависеть от других внешних условий. Детали проверки подписи зависят от используемого алгоритма (см. раздел 7.7).

Получатель не должен полагаться на значения хэш-функции сообщений, вычисленные издателем. Если `SignedData.signerInfo` включает `signedAttributes`, то значение хэш-функции содержимого должно быть рассчитано, как описано в разделе 7.4. Чтобы подпись была действительной, значение хэш-функции, вычисленное получателем, должно быть таким же, как значение атрибута `messageDigest`, включенного в подписанные атрибуты `SignedData.signerInfo`.

Если SignedData.signerInfo включает signedAttributes, значение атрибута content-type должно соответствовать значению SignedData.encapContentInfo.eContentType.

7.7 Специфика данных типа «подписанные данные» по ГОСТ Р 34.10-2012

В данном разделе описано использование алгоритмов подписи по ГОСТ Р 34.10-2012 в CMS.

Идентификаторы алгоритма подписи указываются в поле signatureAlgorithm структуры SignerInfo (см. раздел 7.3), вложенной в структуру SignedData. Идентификаторы алгоритма подписи также указываются в поле signatureAlgorithm структуры SignerInfo атрибутов удостоверяющей подписи.

Значения подписи указываются в поле signature структуры SignerInfo, вложенной в структуру SignedData. Значения подписи также указываются в поле подписи SignerInfo атрибутов удостоверяющей подписи.

Идентификаторы объектов АСН.1, параметры, битовое представление в точности совпадает с представлением подписи в сертификате, описанным в [3].

Сертификат, используемый для подписи сообщений, должен содержать расширение KeyUsage, и данное расширение должно содержать флаг digitalSignature.

7.7.1 Специфика использования ГОСТ Р 34.10-2012 с ключом 256 бит

Алгоритм подписи по ГОСТ Р 34.10-2012 с ключом 256 бит используется для формирования ЭП в форме двух 256-битных чисел r и s по алгоритму ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит. Её представление в виде строки октетов состоит из 64 октетов: при этом первые 32 октета содержат число s в представлении big-endian (старший октет записывается первым), а вторые 32 октета содержат число r в представлении big-endian.

```
GostR3410-2012-256-Signature ::= OCTET STRING (SIZE (64))
```

7.7.2 Специфика использования ГОСТ Р 34.10-2012 с ключом 512 бит

Алгоритм подписи по ГОСТ Р 34.10-2012 с ключом 512 бит используется для формирования ЭП в форме двух 512-битных чисел r и s по алгоритму ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит. Её представление в виде строки октетов состоит из 128 октетов: при этом первые 64 октета содержат число s в представлении big-endian (старший октет записывается первым), а вторые 64 октета содержат число r в представлении big-endian.

```
GostR3410-2012-512-Signature ::= OCTET STRING (SIZE (128))
```

8 Содержимое типа «конверт данных»

Содержимое типа «конверт данных» представляет собой зашифрованное содержимое и зашифрованные ключи шифрования содержимого для одного или более получателей. Комбинация зашифрованного содержимого и одного зашифрованного ключа для одного получателя называется цифровым конвертом получателя. Любое содержимое может быть обернуто для произвольного числа получателей, при использовании любого поддерживаемого получателем механизма управления ключами (см. раздел 8.2).

Построение содержимого типа «конверт данных» состоит из следующих шагов:

1. Генерируется случайный ключ шифрования содержимого.
2. Ключ шифрования содержимого зашифровывается для каждого получателя. Детали шифрования зависят от используемого механизма управления ключами (см. раздел 8.4). В сообщениях формата CMS с использованием российских криптографических алгоритмов применяются 2 способа управления ключами:
   - использование симметричного общего ключа, выработанного при помощи ключа проверки ЭП получателя и ключа ЭП отправителя, для шифрования содержимого (см. разделы **Error! Reference source not found.**, 8.2.2);
   - использование ранее выработанного симметричного ключа для шифрования содержимого (см. разделы 8.2.3);
3. Для каждого получателя зашифрованный ключ шифрования содержимого и другая информация, специфичная для получателя, записываются в структуру RecipientInfo (см. раздел 8.2).
4. Содержимое зашифровывается с помощью ключа шифрования содержимого. При шифровании может возникнуть необходимость дополнения данных содержимого до полного блока (см. раздел 8.3).
5. Значение `RecipientInfo` вместе с зашифрованным содержимым для всех получателей записывается в структуру `EnvelopedData` (см. раздел 8.1).

Получатель «распаковывает» содержимое следующим образом: сначала расшифровывается один из зашифрованных ключей шифрования содержимого, затем с помощью полученного ключа расшифровывается само содержимое.

8.1 Тип EnvelopedData

Следующий идентификатор определяет тип содержимого `EnvelopedData`:

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

Тип `EnvelopedData` в формате АСН.1 представляется следующим образом:

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

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

EncryptedContentInfo ::= SEQUENCE
{
     contentType                  ContentType,
     contentEncryptionAlgorithm
     ContentEncryptionAlgorithmIdentifier,
     encryptedContent[0]          IMPLICIT EncryptedContent OPTIONAL
}
ContentType ::= OBJECT IDENTIFIER
ContentEncryptionAlgorithmIdentifier ::= SEQUENCE
{
     encryptionAlgorithmOID   OBJECT IDENTIFIER,
     parameters               Gost3412-15-Encryption-Parameters
}
Gost3412-15-Encryption-Parameters ::= SEQUENCE
{
     ukm  OCTET STRING
}
EncryptedContent ::= OCTET STRING
UnprotectedAttributes ::= SET SIZE (1..MAX) OF Attribute
```

Типы `CMSVersion`, `OriginatorInfo`, `AlgorithmIdentifier`, `Attribute` определены в разделе 14. Тип `RecipientInfo` определен в разделе 8.2.

Поля структуры `EnvelopedData` имеют следующий смысл:

`version` - номер версии синтаксиса. Версия синтаксиса должна определяться согласно [1], раздел 6.1.

originatorInfo - поле, содержащее информацию об отправителе; данное поле присутствует в структуре, если это требует алгоритм согласования ключей: в случае использования алгоритма согласования ключей на основе протокола Диффи-Хеллмана с фиксированными параметрами данное поле обязательно должно присутствовать в структуре. В этом случае поле содержит информацию о сертификате, ключ которого используется для согласования ключей. Поле может содержать информацию о нескольких сертификатах и CAC. Наличие в зашифрованном сообщении (в том числе и защищенном имитовставкой) поля originatorInfo не является подтверждением авторства данного сообщения.

recipientInfos – поле, содержащее список информации о каждом из получателей; данный список не может быть пустым.

encryptedContentInfo - зашифрованное содержимое;

unprotectedAttrs - необязательный набор незашифрованных атрибутов;

Поля структуры EncryptedContentInfo имеют следующий смысл:

contentType - тип содержимого.

contentEncryptionAlgorithm - определение алгоритма зашифрования содержимого и его параметры; процесс зашифрования содержимого описывается в разделе 8.3; для всех получателей используется один и тот же алгоритм зашифрования содержимого.

encryptedContent - необязательное поле, результат зашифрования содержимого; если поле отсутствует, то предполагается, что значение будет получено другим способом.

Так как поле recipientInfos расположено до поля encryptedContentInfo, то значение EnvelopedData может быть обработано за один проход.

8.2 Тип RecipientInfo

Информация о каждом из получателей хранится в структуре RecipientInfo. Структура RecipientInfo имеет разные форматы в зависимости от используемого механизма управления ключами (подробнее см. [1]). Любые механизмы управления ключами могут быть использованы для одного зашифрованного содержимого. Зашифрованный ключ шифрования содержимого передается одному или более получателям.

Все реализации должны корректно обрабатывать неподдерживаемые алгоритмы.

Для российских криптографических алгоритмов должны поддерживаться алгоритмы согласования ключей и симметричное шифрование ключей, как это указано в полях ktri, kari и kekri, соответственно. Так как различные получатели могут поддерживать различные механизмы управления ключами, и будущие спецификации

могут определить дополнительные механизмы управления ключами, то все реализации должны корректно обрабатывать:

- все возможные варианты значения поля `RecipientInfo`
- все возможные версии реализаций для значений поля `RecipientInfo`
- все нереализованные и неизвестные варианты значения поля `RecipientInfo` для `OtherRecipientInfo`.

Тип `RecipientInfo` в формате АСН.1 представляется следующим образом:

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

Типы `KeyTransRecipientInfo`, `KeyAgreeRecipientInfo`, `KEKRecipientInfo` определены в разделах **Error! Reference source not found.**, 8.2.2, 8.2.3, соответственно. Типы `PasswordRecipientinfo` и `OtherRecipientInfo` не рассматриваются в данной спецификации.

Совместимые реализации, оперирующие сообщениями с типом данных «конверт данных», должны поддерживать следующий функционал при зашифровании и расшифровании:

- Обработка шифрованных сообщений с использованием протокола Диффи-Хеллмана с разовыми параметрами;
- Закодирование и раскодирование информации о получателях шифрованного сообщения в виде структур типа `KeyTransRecipientInfo`;
- Обработка шифрованных сообщений без имитовставки.

Поддержка следующих типов шифрованных сообщений является опциональной и остается на усмотрение разработчика:

- Сообщения с использованием протокола Диффи-Хеллмана с фиксированными параметрами;
- Сообщения с использованием структур типа KeyAgreeRecipientInfo и KEKRecipientInfo;
- Сообщения с имитозащитой.

8.2.1 Тип KeyTransRecipientInfo

Информация о каждом пользователе, использующем механизм управления ключами на основе протокола Диффи-Хеллмана с разовыми параметрами, хранится в структуре `KeyTransRecipientInfo`. Каждый экземпляр структуры `KeyTransRecipientInfo` содержит ключ шифрования содержимого, зашифрованный для одного из получателей.

Тип `KeyTransRecipientInfo` в формате АСН.1 представляется следующим образом:

```
KeyTransRecipientInfo ::= SEQUENCE
{
     version                 CMSVersion,
     rid                     RecipientIdentifier,
     keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
     encryptedKey            GostR3410-KeyTransport
}
RecipientIdentifier ::= CHOICE
{
     issuerAndSerialNumber        IssuerAndSerialNumber,
     subjectKeyIdentifier[0]      SubjectKeyIdentifier
}
SubjectKeyIdentifier ::= OCTET STRING
KeyEncryptionAlgorithmIdentifier ::= SEQUENCE
{
     algorithm      OBJECT IDENTIFIER,
     parameters     GostR3410-12-KEG-Parameters
}
GostR3410-12-KEG-Parameters ::= SEQUENCE
{
     algorithm OBJECT IDENTIFIER
}
GostR3410-KeyTransport ::= SEQUENCE
{
     encryptedKey        OCTET STRING,
     ephemeralPublicKey  SubjectPublicKeyInfo,
     ukm                 OCTET STRING
```

```
}
SubjectPublicKeyInfo ::=  SEQUENCE
{
     algorithm           AlgorithmIdentifier,
     subjectPublicKey    BIT STRING
}
```

Поля структуры `KeyTransRecipientInfo` имеют следующий смысл:

`version` - номер версии синтаксиса; в зависимости от значения поля `rid` типа `RecipientIdentifier` номер версии равен или 0, или 2:

- если параметр `rid` равен `issuerAndSerialNumber`, то версия равна 0,
- если параметр `rid` равен `subjectKeyIdentifier[0]`, то версия равна 2.

`rid` - определение сертификата получателя или непосредственно самого ключа проверки ЭП, который был использован отправителем для защиты ключа шифрования содержимого. Ключ шифрования содержимого зашифрован с использованием ключа проверки ЭП получателя. Тип `RecipientIdentifier` предоставляет два варианта указания сертификата получателя и тем самым ключа проверки ЭП получателя. Сертификат ключа получателя должен содержать ключ проверки ЭП пригодный для использования в данном механизме управления ключами. Таким образом, для российских криптографических алгоритмов сертификат стандарта X.509 получателя версии 3, который содержит в расширении `KeyUsage` установленный флаг `keyAgreement`. Альтернативное значение `issuerAndSerialNumber` идентифицирует получателя по выделенному имени издателя и порядковому номеру сертификата. Для ссылки на сертификат стандарта X.509 должен использоваться идентификатор ключа, соответствующий расширению `SubjectKeyIdentifier` в сертификате. При использовании ссылки на другие форматы сертификатов документ, определяющий формат сертификата и его использование в CMS, должен включать сведения о соответствии идентификатора ключа соответствующему полю сертификата. Для обработки на стороне получателей реализация должна поддерживать оба способа ссылок на сертификат. Для обработки на стороне отправителя, реализация должна поддерживать хотя бы одну альтернативу.

`keyEncryptionAlgorithm` - определение алгоритма зашифрования ключа шифрования содержимого и его параметров; процесс зашифрования описан и значения параметров определены в разделе 8.4.

`encryptedKey` - поле представляет собой структуру `GostR3410-KeyTransport`.

Поля структуры `GostR3410-KeyTransport` имеют следующий смысл:

`encryptedKey` – зашифрованные по алгоритму KExp15 (см. [4]) ключ шифрования и имитовставка (см. раздел 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт - имитовставку. В зависимости от алгоритма n принимает следующие значения (см. раздел8.4.1):

- для алгоритма `id-gostr3412-2015-magma-wrap-kexp15` n=8,
- для алгоритма `idid-gostr3412-2015-kuznyechik-wrap-kexp15` n=16.

`ephemeralPublicKey` - эфемерный ключ проверки ЭП отправителя, данный параметр является обязательным.

`ukm` - ключевой материал пользователя (UKM), параметр является обязательным и должен содержать 32 октета.

Поля структуры `SubjectPublicKeyInfo` имеют следующий смысл:

`algorithm` – определение алгоритма ключа проверки ЭП;

`subjectPublicKey` – значение ключа проверки ЭП.

Типы `CMSVersion`, `IssuerAndSerialNumber` и `AlgorithmIdentifier` определены в разделе 14.

8.2.2 Тип KeyAgreeRecipientInfo

Информация о получателе с использованием протокол согласования ключей хранится в структуре `KeyAgreeRecipientInfo`. Каждый экземпляр `KeyAgreeRecipientInfo` содержит ключ шифрования содержимого для одного или нескольких получателей, которые используют тот же алгоритм согласования и одинаковые его параметры.

```
KeyAgreeRecipientInfo ::= SEQUENCE
{
     version                 CMSVersion,
     originator[0]           EXPLICIT OriginatorIdentifierOrKey,
     ukm[1]                  EXPLICIT UserKeyingMaterial,
     keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
     recipientEncryptedKeys  RecipientEncryptedKeys
}
OriginatorIdentifierOrKey ::= CHOICE
{
     issuerAndSerialNumber   IssuerAndSerialNumber,
     subjectKeyIdentifier[0] SubjectKeyIdentifier,
     originatorKey[1]        OriginatorPublicKey
}
SubjectKeyIdentifier ::= OCTET STRING

OriginatorPublicKey ::= SEQUENCE
```

```
{
     algorithm AlgorithmIdentifier,
     publicKey OCTET STRING
}
UserKeyingMaterial ::= OCTET STRING
KeyEncryptionAlgorithmIdentifier ::= SEQUENCE
{
     algorithm      OBJECT IDENTIFIER,
     parameters     GostR341012KEGParameters
}
GostR341012KEGParameters ::= SEQUENCE
{
     algorithm OBJECT IDENTIFIER
}
RecipientEncryptedKeys ::= SEQUENCE OF RecipientEncryptedKey
RecipientEncryptedKey ::= SEQUENCE
{
     rid            KeyAgreeRecipientIdentifier,
     encryptedKey   OCTET STRING
}
KeyAgreeRecipientIdentifier ::= CHOICE
{
     issuerAndSerialNumber    IssuerAndSerialNumber,
     rKeyId[0]                IMPLICIT RecipientKeyIdentifier
}
RecipientKeyIdentifier ::= SEQUENCE
{
     subjectKeyIdentifier     SubjectKeyIdentifier,
     date                     GeneralizedTime OPTIONAL,
     other                    OtherKeyAttribute OPTIONAL
}
SubjectKeyIdentifier ::= OCTET STRING
```

Типы `CMSVersion`, `IssuerAndSerialNumber`, `AlgorithmIdentifier`, `OtherKeyAttribute` определены в разделе 14.

Поля структуры `KeyAgreeRecipientInfo` имеют следующий смысл:

`version` - номер версии синтаксиса; должен быть всегда равен 3;

`originator` - один из трех вариантов определения ключа проверки ЭП отправителя. Отправитель использует соответствующий ключ ЭП и ключ проверки ЭП получателя для выработки ключа парной связи. Ключ шифрования содержимого зашифровывается на ключе парной связи. Реализации должны поддерживать все три варианта для определения ключа проверки ЭП отправителя:

- вариант `issuerAndSerialNumber` идентифицирует сертификат отправителя и тем самым ключ проверки ЭП отправителя по выделенному имени издателя и серийному номеру сертификата;
- вариант `subjectKeyIdentifier` идентифицирует сертификат отправителя и тем самым ключ проверки ЭП отправителя по идентификатору ключа; при использовании ссылок на сертификат стандарта X.509 используется идентификатор ключа, соответствующий X.509 расширению `subjectKeyIdentifier`; при использовании ссылок на другие форматы сертификата документы, которые определяют формат сертификата и его использование в CMS, должны включать сведения о соответствующем поле сертификата;
- вариант `originatorKey` включает идентификатор алгоритма и ключ проверки ЭП отправителя; этот вариант допускает анонимность отправителя, так как ключ проверки ЭП не подтвержден;

`ukm` - пользовательский ключевой материал (UKM), данный параметр является обязательным и должен содержать 32 октета;

`keyEncryptionAlgorithm` - определение алгоритма зашифрования ключа шифрования содержимого и его параметров; процесс зашифрования описан и значения параметров определены в разделе 8.4.

`recipientEncryptedKeys` - список структур типа `RecipientEncryptedKey`. Каждый экземпляр структуры `RecipientEncryptedKey` включает в себя идентификатор получателя типа `KeyAgreeRecipientIdentifier` и параметр `encryptedKey` для одного или нескольких получателей.

Параметр типа `KeyAgreeRecipientIdentifier` представляет собой выбор с двумя вариантами указания сертификата получателя, и тем самым ключа проверки ЭП получателя, который использовался отправителем для выработки парного ключа шифрования:

- вариант `issuerAndSerialNumber` идентифицирует сертификат по выделенному имени и серийному номеру;
- вариант `RecipientKeyIdentifier` описан ниже;

`encryptedKey` – зашифрованные по алгоритму KExp15 (см. [4]) ключ шифрования и имитовставка (см. раздел 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт - имитовставку. В зависимости от алгоритма n принимает следующие значения (см. раздел8.4.1):

- для алгоритма `id-gostr3412-2015-magma-wrap-kexp15` n=8,
- для алгоритма `idid-gostr3412-2015-kuznyechik-wrap-kexp15` n=16.

Сертификат ключа получателя должен содержать ключ проверки ЭП пригодный для использования в данном механизме управления ключами. Таким образом, для российских криптографических алгоритмов сертификат стандарта X.509 получателя версии 3, который содержит в расширении `KeyUsage` установленный флаг `keyAgreement`.Ключ шифрования содержимого шифруется ключом парной связи.

Поля типа `RecipientKeyIdentifier` имеют следующий смысл:

`subjectKeyIdentifier` - определение сертификата получателя при помощи идентификатора ключа; при использовании ссылок на сертификат стандарта X.509 используется идентификатор ключа, соответствующий X.509 расширению `subjectKeyIdentifier`; при использовании ссылок на другие форматы сертификата документы, которые определяют формат сертификата и его использование в CMS, должны включать сведения о соответствующем поле сертификата;

`date` - дата, необязательное поле; если поле присутствует, то дата определяет какой из ранее распределенных получателем UKM был использован отправителем;

`other` - дополнительная информация, используемая получателем для нахождения публичного ключевого материала UKM, используемого отправителем; данный параметр не является обязательным.

8.2.3 Тип KEKRecipientInfo

Информация о получателе с использованием ранее распределенных симметричных ключей представлена в структуре `KEKRecipientInfo`. Каждая структура содержит ключ шифрования содержимого для одного или нескольких получателей, которым ранее были распределены ключи зашифрования ключей шифрования содержимого.

```
KEKRecipientInfo ::= SEQUENCE
{
     version              CMSVersion,
     kekid                KEKIdentifier,
     keyEncryptionAlgorithm   KeyEncryptionAlgorithmIdentifier,
     encryptedKey         OCTET STRING
}
KEKIdentifier ::= SEQUENCE
{
     keyIdentifier  OCTET STRING,
     date           GeneralizedTime OPTIONAL,
```

```
    other          OtherKeyAttribute OPTIONAL
}
KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
```

Типы `CMSVersion`, `AlgorithmIdentifier`, `OtherKeyAttribute` определены в разделе 14.

Поля типа `KEKRecipientInfo` имеют следующий смысл:

`version` - номер версии синтаксиса; должен всегда равняться 4;

`kekid` - идентификатор симметричного ключа, который ранее был передан отправителю и одному или нескольким получателям;

`keyEncryptionAlgorithm` - определение алгоритма зашифрования ключа шифрования содержимого и его параметров; процесс зашифрования описан и значения параметров определены в разделе 8.4.;

`encryptedKey` – зашифрованные по алгоритму KExp15 (см. [4]) ключ шифрования и имитовставка (см. раздел 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт - имитовставку. В зависимости от алгоритма n принимает следующие значения (см. раздел8.4.1):

- для алгоритма `id-gostr3412-2015-magma-wrap-kexp15` n=8,
- для алгоритма `idid-gostr3412-2015-kuznyechik-wrap-kexp15` n=16.

Поля типа `KEKIdentifier` имеют следующий смысл:

`keyIdentifier` - идентификатор ключа зашифрования ключа, который ранее был передан отправителю и одному или нескольким получателям;

`date` - дата, необязательное поле; если поле присутствует, то дата определяет единственный ключ зашифрования ключа шифрования содержимого, который ранее был распределен;

`other` - дополнительная информация, используемая для определения ключа шифрования ключа отправителя; необязательное поле.

8.3 Процесс зашифрования содержимого

Для требуемого алгоритма ключ шифрования содержимого генерируется случайным образом. При использовании российских криптографических алгоритмов процедура дополнения содержимого не производится. Операция зашифрования отображает произвольную строку октетов (данные) в другую строку октетов (зашифрованный текст) с использованием ключа шифрования. Зашифрованные данные включаются в поле структуры `EnvelopedData`: `OCTET STRING EnvelopedData.encryptedContentInfo.encryptedContent`.

8.3.1 Параметры шифрования сообщений

В данном разделе изложены рекомендации, используемые при реализации CMS с поддержкой шифрования содержимого согласно ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015.

Идентификаторы алгоритма шифрования содержимого указываются в следующих полях:

- `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm`,
- `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm`

(определение структуры `EncryptedData` см. раздел 10).

Для шифрования содержимого могут использоваться следующие идентификаторы:

```
id-gostr3412-2015-magma-ctracpkm OBJECT IDENTIFIER ::=
{
   iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
   algorithms(1) cipher(5) gostr3412-2015-magma(1) mode-ctracpkm(1)
}
```

При использовании идентификатора `id-gostr3412-2015-magma-ctracpkm` шифрование производится с помощью блочного шифра ГОСТ Р 34.12-2015 «Магма» в режиме CTR-ACPKM с параметром s=64 (см. [4])

```
id-gostr3412-2015-magma-ctracpkm-omac OBJECT IDENTIFIER ::=
{
  iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
  algorithms(1) cipher(5) gostr3412-2015-magma(1) mode-ctracpkm-omac(2)
}
```

При использовании идентификатора `id-gostr3412-2015-magma-ctracpkm-omac` вычисление имитовставки осуществляется с помощью шифра ГОСТ Р 34.12-2015 «Магма» в режиме выработки имитовставки ГОСТ Р 34.13-2015 (размер имитовставки – 64 бита), а шифрование производится с помощью шифра ГОСТ Р 34.12-2015 «Магма» в режиме CTR-ACPKM с параметром s=64 (см. [4]).

```
id-gostr3412-2015-kuznyechik-ctracpkm OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms(1) cipher(5) gostr3412-2015-kuznyechik(2) mode-ctracpkm(1)
}
```

При использовании идентификатора `id-gostr3412-2015-kuznyechik-ctracpkm` шифрование производится с помощью блочного шифра ГОСТ Р 34.12-2015 «Кузнечик» в режиме CTR-ACPKM с параметром s=128 (см. [4])

```
id-gostr3412-2015-kuznyechik-ctracpkm-omac OBJECT IDENTIFIER ::=
{
```

```
    iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
    algorithms(1) cipher(5) gostr3412-2015-kuznyechik(2) mode-ctracpkm-omac(2)
}
```

При использовании идентификатора `id-gostr3412-2015-kuznyechik-ctracpkm-omac` вычисление имитовставки осуществляется с помощью шифра ГОСТ Р 34.12-2015 «Кузнечик» в режиме выработки имитовставки ГОСТ Р 34.13-2015 (размер имитовставки – 128 бит), а шифрование производится с помощью шифра ГОСТ Р 34.12-2015 «Кузнечик» в режиме CTR-ACPKM с параметром s=128 (см. [4]).

Алгоритмы шифрования используются для шифрования содержимого, указанного в полях:

- `EnvelopedData.EncryptedContentInfo.encryptedContent`;
- `EncryptedData.EncryptedContentInfo.encryptedContent`.

В качестве дополнительных параметров используются параметры из полей `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm` и `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`, соответственно (см. раздел 8.1).

8.3.2 Процесс формирования и разбора содержимого

Для алгоритмов `id-gostr3412-2015-magma-ctracpkm` и `id-gostr3412-2015-kuznyechik-ctracpkm`:

- Зашифрование содержимого осуществляется следующим способом:
  1. С использованием ключа шифрования содержимого $\boldsymbol{K_{me}}$ в зависимости от заданного алгоритма шифрования происходит зашифрование содержимого с начальным значением $\boldsymbol{IV}$, равным соответствующим полям:
     - `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;
     - `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`.
  2. Ключ шифрования содержимого $\boldsymbol{K_{me}}$ зашифровывается в соответствии с выбранным механизмом управления ключами и устанавливается в соответствующее поле.
- Расшифрование содержимого осуществляется следующим способом:
  1. Извлекается и расшифровывается ключ шифрования содержимого $\boldsymbol{K_{me}}$ (см. раздел 8.4.2);
  2. С использованием ключа $\boldsymbol{K_{me}}$ осуществляется расшифрование содержимого с начальным значением $\boldsymbol{IV}$, равным соответствующим полям:
     - `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;

- `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`.

Для алгоритмов `id-gostr3412-2015-magma-ctracpkm-omac` и `id-gostr3412-2015-kuznyechik-ctracpkm-omac`:

- Зашифрование содержимого осуществляется следующим способом:
  1. Из ключа $\boldsymbol{K_{me}}$ с использованием алгоритма KDF_TREE_GOSTR3411_2012_256, описанного в Р 50.1.113–2016, вырабатывается 2 ключа:
     - ключ шифрования сообщения $\boldsymbol{K(1)}$;
     - ключ вычисления имитовставки $\mathbf{K(2)}$;
  2. С использованием ключа вычисления имитовставки $\mathbf{K(2)}$, выработанного из $\boldsymbol{K_{me}}$, осуществляется вычисление имитовставки для открытого текста содержимого. Результирующее значение имитовставки зашифровывается с использованием ключа шифрования $\boldsymbol{K(1)}$, выработанного из $\boldsymbol{K_{me}}$, с начальным значением $\boldsymbol{IV}$, равным соответствующим полям:
     - `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;
     - `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`.

Зашифрованная имитовставка в виде атрибута помещается в поле незашифрованных атрибутов сообщения (`unprotectedAttrs`) (см. раздел 13.4);

  3. С использованием ключа шифрования $\boldsymbol{K(1)}$, выработанного из $\boldsymbol{K_{me}}$, осуществляется зашифрование содержимого с начальным значением $\boldsymbol{IV}$, равным соответствующим полям:
     - `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;
     - `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;
  4. Ключ шифрования содержимого $\boldsymbol{K_{me}}$ зашифровывается в соответствии с выбранным механизмом управления ключами и устанавливается в соответствующее поле.
- Расшифрование содержимого осуществляется следующим способом:
  1. Извлекается и расшифровывается ключ шифрования содержимого $\boldsymbol{K_{me}}$ (см. раздел 8.4.2);
  2. Из ключа $\boldsymbol{K_{me}}$ с использованием алгоритма KDF_TREE_GOSTR3411_2012_256, описанного в Р 50.1.113–2016, вырабатывается 2 ключа:
     - ключ шифрования сообщения $\boldsymbol{K(1)}$;
     - ключ вычисления имитовставки $\boldsymbol{K(2)}$;
  3. С использованием ключа шифрования $\boldsymbol{K(1)}$, выработанного из $\boldsymbol{K_{me}}$, осуществляется расшифрование содержимого с начальным значением $\boldsymbol{IV}$, равным соответствующим полям:
     - `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm`;

- EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm;

4. С использованием ключа шифрования $\boldsymbol{K(1)}$, выработанного из $\boldsymbol{K_{me}}$,осуществляется расшифрование значения имитовставки сообщения из набора незашифрованных атрибутов (unprotectedAttrs) сообщения (см. раздел 13.4) и обозначается как mac-text.
5. С использованием ключа вычисления имитовставки $\boldsymbol{K(2)}$, выработанного из $\boldsymbol{K_{me}}$, осуществляется вычисление имитовставки для расшифрованного текста. Результирующее значение обозначается как mac-count;
6. Полученный mac-text сравнивается с вычисленным mac-count. При несовпадении размеров или значений сообщение считается искаженным.

8.4 Процесс зашифрования ключа

Исходными данными для процесса зашифрования ключа служит значение ключа шифрования содержимого K_{me}. Шифрование ключа содержимого осуществляется на ключе шифрования ключей, вырабатываемом или генерируемом исходя из выбранного механизма управления ключами. Любой из указанных выше механизмов управления ключами может быть использован для каждого получателя содержимого. Операция зашифрования отображает произвольную строку октетов (данные) в другую строку октетов (зашифрованный текст) с использованием ключа шифрования. Зашифрованный ключ включается в поле структуры EnvelopedData.RecipientInfo.EncryptedKey в зависимости от выбранного механизма управления ключами.

8.4.1 Параметры шифрования ключей

В данном разделе изложены рекомендации, используемые при реализации CMS с поддержкой шифрования ключей согласно алгоритмам KExp15 (см. [4]).

Алгоритм зашифрования ключа описывается в поле keyEncryptionAlgorithm.algorithm и должен принимать одно из следующих значений:

```
id-gostr3412-2015-magma-wrap-kexp15 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms(1) wrap(7) gostr3412-2015-magma(1) kexp15(1)
}
id-gostr3412-2015-kuznyechik-wrap-kexp15 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms(1) wrap(7) gostr3412-2015-kuznyechik(2) kexp15(1)
}
```

Для механизмов управления ключами `ktri` и `kari` дополнительными параметрами для алгоритма зашифрования ключа служит структура `GostR3410-12-KEG-Parameters`, расположенная в поле `keyEncryptionAlgorithm.parameters` и принимающая одно из следующих значений:

```
id-tc26-agreement-gost-3410-12-256 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms (1) agreement(6) gost3410-2012-256(1)
}
id-tc26-agreement-gost-3410-12-512 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms (1) agreement(6) gost3410-2012-512(2)
}
```

В зависимости от заданного идентификатора алгоритма согласования (OID) выбираются KEG-256 или KEG-512.

Процедура зашифрования ключа для механизмов управления ключами `ktri` и `kari` следующая:

- В зависимости от механизма управления ключами вырабатываются ключи $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$.
- В зависимости от алгоритма зашифрования ключа происходит зашифрование ключа на ключе $\boldsymbol{KEK_t}$ и с выработкой имитовставки на ключе $\boldsymbol{KIM_t}$ по алгоритму KExp15 (см. [4]).

Процедура расшифрования выполняется аналогично.

8.4.2 Процесс формирования и разбора ключей

8.4.2.1 Процесс формирования и разбора ключей на основе протокола Диффи-Хеллмана с фиксированными параметрами

Для формирования структуры с использованием сертификата ключа проверки ЭП получателя используется следующий алгоритм:

1. Проверяется соответствие параметров ключа проверки ЭП отправителя и ключа проверки ЭП получателя. При несовпадении параметров алгоритм завершается с ошибкой.
2. В качестве ключа $\boldsymbol{K_s}$ принимается ключ ЭП отправителя.
3. Заполняется поле `EnvelopedData.recipientInfos.kari.originator` в зависимости от желаемого способа задания ключа проверки ЭП отправителя (см. раздел 8.2.2). Параметры и битовое представление ключа проверки ЭП в точности совпадают с их представлением в сертификате (см. [3]).

4. Вырабатывается уникальный (случайный) не нулевой $\boldsymbol{UKM}$ длиной 32 октета, кодируется по правилам типа `OCTET STRING` и записывается в поле `EnvelopedData.recipientInfos.kari.ukm`.
5. Для каждого получателя вырабатываются два ключа $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$.с использованием ключа $\boldsymbol{K_s}$, ключа проверки ЭП получателя $\boldsymbol{P_r}$, $\boldsymbol{UKM}$ и алгоритма KEG (см. [5], раздел 6.4.5):

$$KIM_t \,||\, KEK_t = \text{KEG}(K_s, P_r, UKM)$$

6. Вырабатывается случайный симметричный ключ $\boldsymbol{K_m}$, соответствующий алгоритму, указанному в поле `EnvelopedData.encryptedContentInfo.contentEncryptionAlgorithm`.
7. Для каждого получателя осуществляется экспорт ключа $\boldsymbol{K_m}$ на ключах $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием алгоритма экспорта KExp15 (см. [4]), при этом в качестве ключа $\boldsymbol{K_{MAC}^{EXP}}$ принимается ключ $\boldsymbol{KIM_t}$, в качестве ключа $\boldsymbol{K_{ENC}^{EXP}}$ принимается ключ $\boldsymbol{KEK_t}$. В качестве $\boldsymbol{IV}$ используется $\boldsymbol{UKM}$[25…24 + n / 2], где n - длина блока в байтах.
8. Результатом работы алгоритма KExp15 является строка KEXP, содержащая в себе зашифрованный ключ и имитовставку в соответствии с [4].
9. Результирующая строка KEXP записывается в поле `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey.encryptedKey`.

Для получения сессионного ключа K_m из сообщения M с использованием ключа ЭП K_r применяется следующий алгоритм:

1. Извлекается строка KEXP из поля `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey.EncryptedKey`, содержащая в себе зашифрованный ключ $\boldsymbol{K_m^{EXP}}$ и имитовставку согласно [4];
2. Выбираются алгоритм и параметры ключа проверки ЭП отправителя $\boldsymbol{P_s}$ из структуры `EnvelopedData.recipientInfos.kari.originator` либо с помощью сертификата, либо с помощью ключа проверки ЭП. Проверяется соответствие алгоритма и параметров собственного ключа ЭП параметрам ключа проверки ЭП отправителя $\boldsymbol{P_s}$. При несоответствии параметров алгоритм завершается с ошибкой.
3. Вырабатываются два ключа $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием ключа ЭП получателя $\boldsymbol{K_r}$, ключа проверки ЭП отправителя $\boldsymbol{P_s}$, $\boldsymbol{UKM}$ и алгоритма KEG (см. [5], раздел 6.4.5.1):

$$KIM_t \,||\, KEK_t = \text{KEG}(K_r, P_s, UKM)$$

4. Осуществляется импорт и проверка имитозащиты ключа $\boldsymbol{K_m^{EXP}}$ на ключах $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием алгоритма KImp15 (см. [4]). При этом в качестве ключа $\boldsymbol{K_{MAC}^{EXP}}$ принимается ключ $\boldsymbol{KIM_t}$, в качестве ключа $\boldsymbol{K_{ENC}^{EXP}}$ принимается ключ $\boldsymbol{KEK_t}$. Результатом импорта является ключ $\boldsymbol{K_m}$.

8.4.2.2 Процесс формирования и разбора ключей на основе протокола Диффи-Хеллмана с разовыми параметрами

Для формирования структуры с использованием сертификата ключа проверки ЭП получателя используется следующий алгоритм:

1. Проверяется, что все сертификаты получателей либо не содержат расширения `KeyUsage`, либо содержат, и в нем присутствует флаг `keyAgreement`. В последнем случае должен отсутствовать флаг `encipherOnly`.
2. Генерируется случайный эфемерный ключ $\boldsymbol{K_e}$ с использованием алгоритма и параметров ключа получателя. Обозначим $\boldsymbol{P_e}$ точку на эллиптической кривой $\boldsymbol{E}$, определяемую следующим равенством:

$$P_e = [K_e]P = \underbrace{P + \cdots + P}_{k \text{ раз}},$$

   где P фиксированная точка эллиптической кривой E, определяемая значением идентификатора, содержащегося в сертификате ключа проверки ЭП P_r.
3. Эфемерный случайный ключ может создаваться как отдельно для каждого получателя, так и один для любого числа получателей.
4. Заполняется поле `EnvelopedData.recipientInfos.ktri.originator`. Алгоритм и параметры эфемерного ключа проверки ЭП отправителя помещаются в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.ephemeralPublicKey`. Ключ проверки ЭП $\boldsymbol{P_e}$ помещается в поле `ephemeralPublicKey.publicKey`. Идентификаторы объектов АСН.1, параметры, битовое представление в точности совпадает с представлением подписи в сертификате, описанным в [3].
5. Вырабатывается уникальный (случайный) не нулевой $\boldsymbol{UKM}$ длиной 32 октета, кодируется по правилам типа `OCTET STRING` и записывается в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.ukm`.
6. Вырабатываются два ключа $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием ключа $\boldsymbol{K_e}$, ключа проверки ЭП получателя $\boldsymbol{P_r}$, $\boldsymbol{UKM}$ и алгоритма KEG (см. [5], раздел 6.4.5.1):

$$KIM_t \,||\, KEK_t = \text{KEG}(K_e, P_r, UKM)$$

7. Вырабатывается случайный симметричный ключ $\boldsymbol{K_m}$, соответствующий алгоритму в поле `EnvelopedData.encryptedContentInfo.contentEncryptionAlgorithm`.
8. Осуществляется экспорт ключа $\boldsymbol{K_m}$ на ключах $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием алгоритма экспорта KExp15 (см. [4]), при этом в качестве ключа $\boldsymbol{K_{MAC}^{EXP}}$ принимается ключ $\boldsymbol{KIM_t}$, в качестве ключа $\boldsymbol{K_{ENC}^{EXP}}$ принимается ключ $\boldsymbol{KEK_t}$. В качестве $\boldsymbol{IV}$ используется $\boldsymbol{UKM}$[25 … 24 + n / 2], где n - длина блока в байтах.

9. Результатом работы алгоритма KExp15 строка KEXP, содержащая в себе зашифрованный ключ и имитовставку в соответствии с [4].
10. Результирующая строка KEXP записывается в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.EncryptedKey`.

Если сообщение отправляется нескольким получателям с различными параметрами ключа проверки ЭП, то для каждого набора параметров генерируется свой эфемерный ключ и создается своя структура `KeyAgreeRecipientInfo`. Сообщения для нескольких получателей с одинаковыми параметрами ключа проверки ЭП могут использовать один эфемерный ключ.

Для получения сессионного ключа K_m из сообщения M с использованием ключа ЭП K_r применяется следующий алгоритм:

1. Извлекается строка KEXP из поля `EnvelopedData.recipientInfos.ktri.encryptedKey.EncryptedKey`, содержащая в себе зашифрованный ключ $\boldsymbol{K_m^{EXP}}$ и имитовставку согласно [4].
2. Выбираются алгоритм, параметры и ключ проверки ЭП отправителя $\boldsymbol{P_e}$ из структуры `EnvelopedData.recipientInfos.kari.originator` либо с помощью сертификата, либо с помощью ключа проверки ЭП. Проверяется соответствие алгоритма и параметров собственного ключа ЭП параметрам ключа проверки ЭП отправителя ($\boldsymbol{P_e}$). При несоответствии параметров алгоритм завершается с ошибкой.
3. Вырабатываются два ключа $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием ключа ЭП получателя $\boldsymbol{K_r}$, ключа проверки ЭП отправителя $\boldsymbol{P_e}$, $\boldsymbol{UKM}$ и алгоритма KEG (см. [5], раздел 6.4.5.1):

$$KIM_t \,||KEK_t = \text{KEG}(K_r, P_e, UKM)$$

4. Осуществляется импорт и проверка имитозащиты ключа $\boldsymbol{K_m^{EXP}}$ на ключах $\boldsymbol{KEK_t}$ и $\boldsymbol{KIM_t}$ с использованием алгоритма KImp15 (см. [4]). При этом в качестве ключа $\boldsymbol{K_{MAC}^{EXP}}$ принимается ключ $\boldsymbol{KIM_t}$, в качестве ключа $\boldsymbol{K_{ENC}^{EXP}}$ принимается ключ $\boldsymbol{KEK_t}$. Результатом импорта является ключ $\boldsymbol{K_m}$.

9 Содержимое типа «хэшированные данные»

Содержимое типа «хэшированные данные» состоит из содержимого любого типа и результата функции хэширования содержимого.

Обычно тип содержимого «хэшированные данные» используется для контроля целостности содержимого, а результат, как правило, становится входом для содержимого типа «конверт данных».

Для построения содержимого типа «хэшированные данные» выполняют следующие шаги:

1. Определяется алгоритм хэширования и вычисляется значение хэш-функции от содержимого
2. Алгоритм вычисления хэш-функции и значение хэш-функции записываются вместе с содержимым в структуру `DigestData`.

Получатель осуществляет подсчет значения хэш-функции и сравнивает результат вычисления с извлеченным из сообщения значением хэш-функции.

Следующий идентификатор определяет тип «хэшированные данные»:

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

Содержимое типа «хэшированные данные» представляется в виде структуры `DigestData`. Структура `DigestData` в формате АСН.1 представляется следующим образом:

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

Типы `CMSVersion` и `AlgorithmIdentifier` определены в разделе 14. Тип `EncapsulatedContentInfo` определен в разделе **Error! Reference source not found.**.

Поля типа `DigestedData` имеют следующий смысл:

`version` **-** номер версии синтаксиса. Версия синтаксиса должна определяться согласно [1], раздел 7.

`digestAlgorithm` - определение алгоритма хэширования и его параметров. Процесс вычисления хэш-функции аналогичен процессу, описанному в разделе 7.4, за исключением того, что отсутствуют подписанные атрибуты.

`encapContentInfo` - хэшируемое содержимое в соответствии с определением из раздела **Error! Reference source not found.**.

`digest` - результат хэширования.

Порядок следования полей в `digestAlgorithm`, `encapContentInfo`, `digest` позволяет обрабатывать значение `DigestData` за один проход.

9.1 Специфика данных типа «хэшированные данные» по ГОСТ Р 34.11-2012

В данном разделе изложены правила использования алгоритмов хэширования по ГОСТ Р 34.11-2012, применяемые в CMS.

Формат, идентификаторы алгоритмов, содержимое атрибутов полностью соответствуют [2].

Использование хэш-кодов в разных типах содержимого в форматах:

- Хэш-код указывается в поле `digest` структуры `DigestedData` для типа содержимого «хэшированные данные»;
- Хэш-код указывается в подписанном атрибуте `MessageDigest` для типов содержимого, имеющих возможность использования подписанного атрибута;
- Хэш-код является входным параметром для алгоритмов подписей.

9.1.1 Сообщения с длиной хэш-кода 256 бит

т

Алгоритм хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит имеет следующий идентификатор:

```
id-tc26-gost3411-2012-256 OBJECT IDENTIFIER ::=
{
    iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
    algorithms(1) digest(2) gost3411-2012-256(2)
}
```

Для задания поля `digest` структуры `DigestedData` или при наличии подписанного атрибута `MessageDigest` значение хэш-функции сообщения содержит 32 октета:

```
GostR3411-2012-256-Digest ::= OCTET STRING (SIZE (32))
```

Представление значения хэш-кода соответствует значению h алгоритма ГОСТ Р 34.11-2012 в виде строки октетов и записывается в формате little-endian (старший октет справа).

9.1.2 Сообщения с длиной хэш-кода 512 бит

Алгоритм хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит имеет следующий идентификатор:

```
id-tc26-gost3411-2012-512 OBJECT IDENTIFIER ::=
{
```

```
    iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
    algorithms(1) digest(2) gost3411-2012-512(3)
}
```

Для задания поля `digest` структуры `DigestedData` или при наличии подписанного атрибута `MessageDigest` значение хэш-функции сообщения содержит 64-октета:

```
    GostR3411-2012-512-Digest ::= OCTET STRING (SIZE (64))
```

Представление значения хэш-кода соответствует значению h алгоритма ГОСТ Р 34.11-2012 в виде строки октетов и записывается в формате little-endian (старший октет справа).

10 Содержимое типа «зашифрованные данные»

Содержимое типа «зашифрованные данные» состоит из зашифрованного содержимого любого типа. В отличие от содержимого типа «конверт данных» данный тип не содержит ни получателей, ни зашифрованных ключей. Распределение ключей происходит с помощью внешних средств.

Обычно тип «зашифрованные данные» применяется для хранения шифрованных локальных данных.

Следующий идентификатор определяет тип «зашифрованные данные»:

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

Содержимое типа «зашифрованные данные» представляется в виде структуры `EncryptedData`. Структура `EncryptedData` в формате АСН.1 представляется следующим образом:

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

```
     contentEncryptionAlgorithm
     ContentEncryptionAlgorithmIdentifier,
     encryptedContent[0]      IMPLICIT EncryptedContent OPTIONAL
}
ContentType ::= OBJECT IDENTIFIER
ContentEncryptionAlgorithmIdentifier ::= SEQUENCE
{
     encryptionAlgorithmOID   OBJECT IDENTIFIER,
     parameters               Gost3412-15-Encryption-Parameters
}
Gost3412-15-Encryption-Parameters ::= SEQUENCE
{
     ukm  OCTET STRING
}
EncryptedContent ::= OCTET STRING
UnprotectedAttributes ::= SET SIZE (1..MAX) OF Attribute
```

Типы `CMSVersion`, `Attribute` определены в разделе 14.

Поля типа `EncryptedData` имеют следующий смысл:

`version` - номер версии синтаксиса. Версия синтаксиса должна определяться согласно [1], раздел 8.

`encryptedContentInfo` - зашифрованное содержимое, подробнее см. раздел 8.1.

`unprotectedAttrs` - набор незашифрованных атрибутов, является необязательным полем.

10.1 Специфика данных типа «зашифрованные данные» по ГОСТ Р 34.12-2015

Входным параметром для алгоритма шифрования по ГОСТ Р 34.12-2015 служит один ключ, полученный внешним способом. Описания параметров зашифрованного сообщения, алгоритма выработки ключей шифрования содержимого и алгоритма вычисления имитовставки, режима шифрования и др. параметров полностью совпадают с аналогичными их описаниями из раздела 8.3.

11 Содержимое типа «аутентифицированные данные»

Содержимое типа «аутентифицированные данные» состоит из содержимого любого типа, имитовставки, зашифрованных ключей для вычисления имитовставки для одного или более получателей.

Сочетание имитовставки и зашифрованного ключа для вычисления имитовставки необходимо для получателя, чтобы проверить целостность содержимого. Любое содержимое может быть защищено для любого количества получателей.

Процесс создания данных типа «аутентифицированные данные» состоит из следующих шагов:

1. Генерируется случайный ключ для конкретного алгоритма вычисления имитовставки.
2. Ключ для вычисления имитовставки зашифровывается для каждого получателя. Подробности этого процесса зависят от используемого механизма управления ключами (более подробно см. раздел 8.4).
3. Для каждого получателя зашифрованный ключ вычисления имитовставки и другая информация, специфичная для данного получателя, записывается в структуру `RecipientInfo`, определенную в разделе 8.2.
4. Используя ключ для вычисления имитовставки, отправитель вычисляет имитовставку для содержимого. Если отправитель использует дополнительное содержимое для аутентификации (см. раздел 11.2), то вычисляется хэш-код от содержимого, от полученного результата и другой информации вычисляется имитовставкой.

11.1 Тип AuthenticatedData

Следующий идентификатор определяет данные типа «аутентифицированные данные»:

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

Содержимое типа «аутентифицированные данные» представляется в виде структуры AuthenticatedData. Структура AuthenticatedData в формате АСН.1 представляется следующим образом:

```
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
}
RecipientInfos ::= SET SIZE (1..MAX) OF RecipientInfo
MessageAuthenticationCodeAlgorithm ::= AlgorithmIdentifier
DigestAlgorithmIdentifier ::= AlgorithmIdentifier
AuthAttributes ::= SET SIZE (1..MAX) OF Attribute
UnauthAttributes ::= SET SIZE (1..MAX) OF Attribute
MessageAuthenticationCode ::= OCTET STRING
```

Типы `CMSVersion`, `OriginatorInfo`, `AlgorithmIdentifier`, `Attribute` определены в разделе 14. Тип `RecipientInfo` определен в разделе 8.2, тип `EncapsulatedContentInfo` определен в разделе **Error! Reference source not found.**.

Поля структуры `AuthenticatedData` имеют следующий смысл:

`version` - номер версии синтаксиса. Версия синтаксиса должна определяться согласно [1], раздел 9.1.

`originatorInfo` - информация об отправителе, необязательное поле, не используется для российских криптографических алгоритмов.

`recipientInfos` - список информации о каждом из получателей, данный список не может быть пустым (подробнее см. раздел 8.1).

`macAlgorithm` - определение алгоритма вычисления имитовставки и его параметров, которые использует отправитель. Расположение поля `macAlgorithm` позволяет выполнять обработку сообщения за один проход.

`digestAlgorithm` - определение алгоритма вычисления хэш-функции и его параметров, используемых для вычисления значения от вложенного содержимого, если присутствуют аутентифицированные атрибуты. Процесс вычисления значения хэш-функции описан в разделе 7.4. Расположение поля `digestAlgorithm` позволяет выполнять обработку сообщения за один проход. Если присутствует поле `digestAlgorithm`, то поле `authAttrs` также должно присутствовать.

`encapContentInfo` - хэшируемое содержимое в соответствии с определением из раздела **Error! Reference source not found.**.

authAttrs - набор аутентифицируемых атрибутов. Поле не является обязательным, но оно должно присутствовать, если тип содержимого структуры EncapsulatedContentInfo не представляет собой «простые данные». Если поле authAttrs присутствует, то и поле digestAlgorithm также должно присутствовать. Поле authAttrs должно быть представлено в DER-кодировке, даже если остальная часть структуры кодируется в BER-кодировку. Если поле присутствует, то оно должно содержать, как минимум два атрибута:

- content-type - атрибут, определяющийся значением типа содержимого EncapsulatedContentInfo (см. раздел 13.1).
- message-digest - атрибут, содержащий значение хэш-функции от содержимое (см. раздел 13.2).

mac — значение имитовставки;

unauthAttrs — набор неаутентифицируемых атрибутов. Поле не является обязательным.

11.2 Формирование имитовставки

Процесс вычисления имитовставки заключается в вычислении имитовставки или для аутентифицируемого содержимого, или для результата вычисления хэш-функции от аутентифицированного содержимого:

1. Если отсутствуют атрибуты в поле authAttrs то имитовставка вычисляется для содержимого поля encapContentInfo.eContent;

2. Если атрибуты в поле authAttrs присутствуют, то для содержимого encapContentInfo.eContent вычисляется хэш-код и помещается в поле message-digest, а имитовставка вычисляется для всего содержимого authAttrs.

Если поле authAttrs отсутствует, то значение поля encapContentInfo.eContent должно представлять собой строку октетов. Входными параметрами для алгоритма вычисления имитовставки являются только те октеты, которые соответствуют значению eContent; октеты тега и длины не должны использоваться. Отсутствие байтов длины в данных типа «аутентифицированные данные» является преимуществом, так как длина не должна быть заранее известна.

Если поле authAttrs присутствует, то атрибут content-type (см. раздел 13.1) и атрибут message-digest (см. раздел 13.2) должны быть включены в результат вычисления имитовставки, и входным параметром для алгоритма вычисления имитовставки является закодированный атрибут authAttrs в DER-кодировке. Тег IMPLICIT[2] в поле authAttrs не используется для DER-кодирования, для этого используется тег EXPLICIT SET OF. То есть в вычисление имитовставки должен быть включен DER закодированный тег SET OF, а не тег IMPLICIT[2], вместе с длиной и содержимым значения authAttrs.

Процесс вычисления хэш-функции заключается в вычислении значения от аутентифицированного сообщения. Входным параметром для алгоритма вычисления хэш-функции служит значение вложенного содержимого для аутентификации. В частности, параметром является значение `encapContentInfo.eContent`, к которому применяется процесс аутентификации. Только те октеты, которые содержат значение `encapContentInfo.eContent`, являются входом для алгоритма вычисления хэш-функции без включения октетов тега и длины. Это является преимуществом, так как длина аутентифицированного содержимого не должна быть известна заранее. Хотя октеты тега и длины поля `encapContentInfo.eContent` не включены в процесс вычисления хэш-функции, они по-прежнему защищены, но другим способом. Целостность значения поля длины обуславливается криптографическими свойствами функции хэширования, поскольку алгоритмически сложно найти какие-либо два значения поля длины, которые имеют одинаковые значения хэш-функции.

Входными параметрами для алгоритма вычисления имитовставки являются данные, определенные выше, и ключ аутентификации, передаваемый в структуре `recipientInfo`. Подробности вычисления имитовставки зависят от используемого алгоритма. Идентификатор алгоритма вместе со своими параметрами, которые задает отправитель для алгоритма вычисления имитовставки, передаются в поле `macAlgorithm`. Имитовставка, вычисленная отправителем, кодируется в строку октетов и записывается в поле `mac`.

11.3 Формирование имитовставки для последующей проверки

Входными параметрами для алгоритма выработки имитовставки для последующего сравнения полученного результата с передаваемым в сообщении служат данные (определяется наличием или отсутствием поля `authAttrs`, см. раздел 11.2) и ключ аутентификации, передаваемый в структуре `recipientInfo`. Подробности вычисления имитовставки зависят от используемого алгоритма (см. раздел 11.4).

Получатель не должен полагаться на любые вычисленные отправителем значения имитовставки и хэш-функции. Содержимое аутентифицируется в соответствии с описанием в разделе 11.2. Если отправитель включил аутентифицируемые атрибуты, то содержимое поля `authAttrs` определяется в соответствии с разделом 11.2. Получатель должен вычислить значение хэш-функции от сообщения и убедиться в его совпадении со значением `message-digest` в составе `authAttrs`. Для успешной аутентификации значение имитовставки, вычисленной получателем, должно совпадать с имитовставкой в поле `mac`.

11.4 Специфика данных типа «аутентифицированные данные» по ГОСТ Р 34.11-2012

В данном разделе изложены соглашения, используемые при реализации CMS с поддержкой кода аутентификации сообщения, согласно Р 50.1.113-2016.

Идентификаторы алгоритма указываются в поле `AuthenticatedData.macAlgorithm`.

Значения кода аутентификации указываются в поле `AuthenticatedData.mac`.

Ключ аутентификации генерируется случайным образом и должен быть защищен на основе алгоритма согласования ключей, изложенного в разделе 8.2.

11.4.1 Сообщения с длиной хэш-кода 256 бит

В основе функции HMAC_GOSTR3411_2012_256 лежит функция хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит (см. [4]).

В настоящем документе для данного алгоритма указан следующий идентификатор объекта (OID):

```
id-tc26-hmac-gost-3411-2012-256 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms (1) mac(4) gost3411-2012-256(1)
}
```

11.4.2 Сообщения с длиной хэш-кода 512 бит

В основе функции HMAC_GOSTR3411_2012_512 лежит функция хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит (см. [4]).

В настоящем документе для данного алгоритма указан следующий идентификатор объекта (OID):

```
id-tc26-hmac-gost-3411-2012-512 OBJECT IDENTIFIER ::=
{
     iso(1) member-body(2) ru(643) rosstandart(7) tc26(1)
     algorithms (1) mac(4) gost3411-2012-512(2)
}
```

12 Вопросы безопасности

Следует учитывать, что аутентификация содержимого обеспечивается использованием протокола Диффи-Хеллмана с фиксированными параметрами для содержимого типов «конверт данных» и «аутентифицированные данные». Для обеспечения аутентификации содержимого типа «конверт данных» с использованием протокола Диффи-Хеллмана с разовыми параметрами и для содержимого типа «зашифрованные данные» необходимо предварительно оборачивать содержимое указанных типов в содержимое типа «подписанные данные» или в содержимое типа «аутентифицированные данные» с использованием протокола Диффи-Хеллмана с фиксированными параметрами.

При использовании протокола Диффи-Хеллмана с фиксированными параметрами не рекомендуется использовать представление ключа проверки ЭП отправителя в виде чистого ключа (`originatorKey[1] OriginatorPublicKey` в структуре из раздела 8.2.2), так как данный вариант не защищён от подмены отправителя.

13 Полезные атрибуты

В данном разделе определяются атрибуты, которые могут использоваться с содержимым типа

- «подписанные данные»,
- «конверт данных»,
- «зашифрованные данные»,
- «аутентифицированные данные».

13.1 Атрибут content-type

Атрибут `content-type` определяет тип содержимого `ContentInfo` в содержимом типов «подписанные данные» или «аутентифицированные данные». Атрибут `content-type` должен присутствовать всякий раз, когда подписанные атрибуты присутствуют в данных типа «подписанные данные» или аутентифицированные атрибуты присутствуют в данных типа «аутентифицированные данные». Значение атрибута `content-type` должно соответствовать значению поля `encapContentInfo.eContentType` в данных обоих типов.

Тип атрибута `content-type` должен быть подписанным или аутентифицированным атрибутом и не должен быть неподписанным атрибутом, не аутентифицированным атрибутом, или незащищенным атрибутом.

Следующий идентификатор определяет атрибут `content-type`:

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

Атрибут `content-type` в формате АСН.1 представлен типом `ContentType`:

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

Так как синтаксис определяет `SET OF AttributeValue`, то атрибут `content-type` должен содержать единственное значение; ноль или несколько экземпляров `AttributeValue` не допускается.

Синтаксис `SignedAttributes` и `AuthAttributes` определяются как `SET OF Attributes`. Структура `SignedAttributes` в `SignerInfo` не должна включать несколько экземпляров атрибута `content-type`. Аналогично, структура `AuthAttributes` в `AuthenticatedData` не должна включать несколько экземпляров атрибута `content-type`.

13.2 Атрибут message-digest

Атрибут `message-digest` содержит хэш-код от подписываемого содержимого из поля `encapContentInfo.eContent` в данных типа «подписанные данные» (см. раздел 7) или в данных типа «аутентифицированные данные» (см. раздел 11). Для данных типа «подписанные данные» значение хэш-функции вычисляется с использованием алгоритма хэширования подписывающего. Для данных типа «аутентифицированные данные» значение хэш-функции вычисляется с использованием алгоритма хэширования отправителя.

В данных типа «подписанные данные» подписанный атрибут `message-digest` должен присутствовать, если присутствуют любые подписанные атрибуты. В данных типа «аутентифицированные данные» аутентифицированный атрибут `message-digest` должен присутствовать, если присутствует любой аутентифицированный атрибут.

Атрибут `message-digest` должен быть подписанным атрибутом или аутентифицированным атрибутом и не должен быть не подписанным атрибутом, не аутентифицированным атрибутом, или незащищенным атрибутом.

Следующий идентификатор определяет `message-digest` атрибут

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

Атрибут `message-digest` в формате АСН.1 представлен типом `MessageDigest`:

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

Атрибут message-digest должен содержать единственное значение, хотя синтаксис определяет SET OF AttributeValue. Он не должен быть пустым или содержать несколько экземпляров AttributeValue.

Синтаксис SignedAttributes и синтаксис AuthAttributes определяют SET OF Attribute. Поле SignedAttributes в структуре SignerInfo должно включать только один экземпляр атрибута message-digest. Аналогично, поле AuthAttributes в структуре AuthenticatedData должно включать только один экземпляр атрибута message-digest.

13.3 Атрибут countersignature

Атрибут countersignature указывает одну или несколько подписей на значение поля SignerInfo.SignatureValue, содержащих октеты подписи для данных типа «подписанные данные». Значение хэш-функции для вычисления удостоверяющей подписи формируется из непосредственно октетов самой подписи, не включая в себя октеты тега и длины. Таким образом, атрибут countersignature удостоверяет другую подпись.

Атрибут countersignature должен быть неподписанным атрибутом и не должен быть подписанным атрибутом, аутентифицированным атрибутом, не аутентифицированным атрибутом или незащищенным атрибутом.

Следующий идентификатор определяет удостоверяющую подпись

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

Атрибут countersignature в формате АСН.1 представлен типом Countersignature:

```
Countersignature ::= SignerInfo
```

Значение countersignature имеет такое же значение как SignerInfo для обычных подписей, за исключением того, что:

1. Поле signedAttributes не должно содержать атрибут content-type; для удостоверяющих подписей нет содержимого;
2. Поле signedAttributes должно содержать атрибут значения хэш-функции, если он содержит любые другие атрибуты;
3. Входными параметрами для алгоритма вычисления хэш-функции служат октеты поля signatureValue структуры SignerInfo, представленного в DER-кодировке.

Атрибут `countersignature` может содержать несколько значений. Синтаксис определяет `SET OF AtrtributeValue` и должен содержать один или несколько экземпляров `AttributeValue`.

Синтаксис атрибута `UnsignedAttributes` определяется как `SET OF Attributes`. Атрибут `UnsignedAttributes` в `signerInfo` может содержать несколько экземпляров атрибута `countersignature`.

Атрибут `countersignature`, так как он имеет тип `SignerInfo`, сам по себе может содержать атрибут `countersignature`. Таким образом, можно построить сколь угодно длинную серию из атрибутов `countersignature`.

13.4 Атрибут content-mac

Атрибут `content-mac` содержимого содержит зашифрованное значение имитовставки, выработанное от незашифрованного значения содержимого сообщения. Формат атрибута представлен в следующей структуре:

```
content-mac ::= SEQUENCE
{
     id-cms-mac-attr          OBJECT IDENTIFIER,
     encryptedMac        OCTET STRING,
     }
```

Следующий идентификатор определяет атрибут `content-mac`:

```
     id-cms-mac-attr OBJECT IDENTIFIER ::=
     {
          iso(1)   member-body(2)   ru(643)   rosstandart(7)   tc26(1)
modules(0)
     cms(6) attributes(1) mac-attibute(1)
     }
```

14 Полезные типы

14.1 CMSVersion

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

Данный тип определяет номер версии синтаксиса для последующей совместимости с новыми версиями спецификации.

14.2 IssuerAndSerialNumber

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

CertificateSerialNumber ::= INTEGER
```

Тип `IssuerAndSerialNumber` определяет сертификат и тем самым ключ проверки ЭП. Данный тип представляет собой структуру со следующими значениями полей:

`issuer` – имя издателя сертификата.

`serialNumber` – серийный номер сертификата.

14.3 RevocationInfoChoices

```
RevocationInfoChoices ::= SET OF RevocationInfoChoice
RevocationInfoChoice ::= CHOICE
{
     crl             CertificateList,
     other[1]        IMPLICIT OtherRevocationInfoFormat
}
OtherRevocationInfoFormat ::= SEQUENCE
{
     otherRevInfoFormat  OBJECT IDENTIFIER,
     otherRevInfo        ANY DEFINED BY otherRevInfoFormat
}
```

Тип `RevocationInfoChoices` определяет множество источников информации об аннулированных сертификатах. Предполагается, что информации достаточно, чтобы проверить корректность сертификата на отзыв, но эта проверка не обязательна. Тип `RevocationInfoChoice` предоставляет два варианта указания источников информации об аннулированных сертификатах. Первый – `crl` – списки аннулированных сертификатов, `other` – любые другие источники информации.

14.4 AlgorithmIdentifier

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

Данный тип предоставляет собой структуру со следующими значениями полей:

`algorithm` – идентификатор используемого алгоритма.

`parameters` – параметры используемого алгоритма.

14.5 CertificateChoices

```
CertificateChoices ::= CHOICE
{
     certificate         Certificate,
     extendedCertificate[0]   IMPLICIT ExtendedCertificate,Obsolete
     v1AttrCert[1]       IMPLICIT AttributeCertificateV1,Obsolete
     v2AttrCert[2]       IMPLICIT AttributeCertificateV2,
     other[3]            IMPLICIT OtherCertificateFormat
}
OtherCertificateFormat ::= SEQUENCE
{
     otherCertFormat     OBJECT IDENTIFIER,
     otherCert           ANY DEFINED BY otherCertFormat
}
```

Тип `CertificateChoices` определяет различные варианты указания формата сертификата:

`certificate` – информация о сертификате.

`extendedCertificate` – сертификат формата PKCS #6, устарел.

`v1AttrCert` – сертификат версии 1 X.509, устарел.

`v2AttrCert` – сертификат версии 2 X.509.

`other` – другой формат.

Информация о сертификате представляет собой следующую подписанную структуру:

```
Certificate ::= SIGNED SEQUENCE
{
     version[0]              Version DEFAULT 1988,
     serialNumber        SerialNumber,
     signature           AlgorithmIdentifier
     issuer              Name
     validity            Validity,
     subject             Name,
     subjectPublicKeyInfo     SubjectPublicKeyInfo
}
Version ::= INTEGER { 1988(0)}
SerialNumber ::= INTEGER
Validity ::= SEQUENCE
{
     notBefore UTCTime,
     notAfter  UTCTime
}
SubjectPublicKeyInfo ::= SEQUENCE
{
     algorithm AlgorithmIdentifier
     subjectKey     BIT STRING
}
```

Поля структуры `Certificate` имеют следующий смысл:

`version` – версия сертификата;

`serialNumber` – серийный номер;

`signature` – определение алгоритма подписи;

`issuer` - имя издателя сертификата;

`validity` – время жизни сертификата;

`subject` – имя субъекта сертификата;

`subjectPublicKeyInfo` – информация о ключе проверки ЭП издателя сертификата.

14.6 CertificateSet

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

Данный тип определяет набор сертификатов. Предполагается, что множество сертификатов достаточно, чтобы построить цепочки сертификатов от корня или от центра сертификации верхнего уровня. Сертификатов может быть больше, чем это необходимо для построения цепочки сертификатов от двух или более центров сертификации верхнего уровня. Сертификатов может быть меньше, чем это необходимо, если получатели имеют альтернативные способы получения необходимых сертификатов (например, из предыдущего набора сертификатов).

14.7 OriginatorInfo

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

Тип `OriginatorInfo` представляет собой структуру со следующими значениями полей:

`certs` - набор сертификатов. Данный параметр может содержать сертификаты отправителя для различных механизмов управления ключами. Также может содержать атрибутивные сертификаты, ассоциированные с отправителем.

`crls` - списки аннулированных сертификатов. Предполагается, что CAC достаточны, чтобы проверить корректность сертификата на отзыв, но эта проверка не обязательна.

14.8 OtherKeyAttribute

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

Тип `OtherKeyAttribute` определяет синтаксис для включения других атрибутов ключа, которые позволяют пользователю выбирать ключ, используемый отправителем. Тип представляет собой структуру со следующими значениями полей:

`keyAttrId` – идентификатор атрибутов ключа.

`keyAttr` – атрибуты ключа.

14.9 Attribute

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

Тип `Attribute` представляет собой структуру со следующими значениями полей:

`attrType` – тип атрибута.

`attrValues` – значение атрибута.

Библиография

[1] IETF RFC 5652	Р. Хаусли, Синтаксис криптографических сообщений (CMS) (Housley R., Cryptographic Message Syntax (CMS)), RFC 5652, сентябрь 2009
[2] Методические рекомендации ТК 26	Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ 2814789, ГОСТ Р 34.11 и ГОСТ Р 34.10 в криптографических сообщениях формата CMS
[3] Методические рекомендации ТК 26	Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ Р 34.10, ГОСТ Р 34.11 в профиле сертификата и списке отзыва сертификатов (CRL) инфраструктуры открытых ключей X.509. Версия 2
[4] Методические рекомендации ТК 26	Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов блочного шифрования
[5] Методические рекомендации ТК 26	Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)

УДК 681.3.06:006.354 ОКС 35. 040 ОКСТУ 5002 П85

Ключевые слова: криптографические протоколы, аутентификация, пароль, ключ