

---

ФЕДЕРАЛЬНОЕ АГЕНТСТВО  
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

---



РЕКОМЕНДАЦИИ  
ПО СТАНДАРТИЗАЦИИ      Р 1323565.1.025—  
2019

---

Информационная технология

**КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ**

**Форматы сообщений, защищенных  
криптографическими методами**

Издание официальное



Москва  
Стандартинформ  
2019

## Предисловие

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

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 29 августа 2019 г. № 593-ст

4 ВВЕДЕНЫ ВПЕРВЫЕ

*Правила применения настоящих рекомендаций установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящим рекомендациям публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а 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](http://www.gost.ru)).*

© Стандартинформ, оформление, 2019

В Российской Федерации настоящие рекомендации не могут быть воспроизведены, тиражированы и распространены в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

## Содержание

1 Область применения . . . . .	1
2 Нормативные ссылки . . . . .	1
3 Термины, определения и обозначения . . . . .	2
3.1 Термины и определения . . . . .	2
3.2 Обозначения . . . . .	2
4 Общее описание . . . . .	3
5 Общий синтаксис . . . . .	3
6 Содержимое типа «простые данные» . . . . .	4
7 Содержимое типа «подписанные данные» . . . . .	4
7.1 Тип SignedData . . . . .	5
7.2 Тип EncapsulatedContentInfo . . . . .	6
7.3 Тип SignerInfo . . . . .	6
7.4 Процесс вычисления значения хэш-функции . . . . .	7
7.5 Процесс формирования подписи . . . . .	8
7.6 Процесс проверки подписи . . . . .	8
7.7 Специфика данных типа «подписанные данные» . . . . .	8
8 Содержимое типа «конверт данных» . . . . .	9
8.1 Тип EnvelopedData . . . . .	9
8.2 Тип RecipientInfo . . . . .	10
8.3 Процесс зашифрования содержимого . . . . .	15
8.4 Процесс зашифрования ключа . . . . .	18
9 Содержимое типа «хэшированные данные» . . . . .	21
9.1 Тип DigestedData . . . . .	21
9.2 Специфика данных типа «хэшированные данные» . . . . .	21
10 Содержимое типа «зашифрованные данные» . . . . .	22
10.1 Тип EncryptedData . . . . .	22
10.2 Специфика данных типа «зашифрованные данные» . . . . .	23
11 Содержимое типа «аутентифицированные данные» . . . . .	23
11.1 Тип AuthenticatedData . . . . .	24
11.2 Формирование имитовставки . . . . .	25
11.3 Формирование имитовставки для последующей проверки . . . . .	25
11.4 Специфика данных типа «аутентифицированные данные» . . . . .	26
12 Вопросы безопасности . . . . .	26
13 Полезные атрибуты . . . . .	26
13.1 Атрибут content-type . . . . .	27
13.2 Атрибут message-digest . . . . .	27
13.3 Атрибут countersignature . . . . .	28
13.4 Атрибут content-mac . . . . .	28
14 Полезные типы . . . . .	29
14.1 Тип CMSVersion . . . . .	29
14.2 Тип IssuerAndSerialNumber . . . . .	29
14.3 Тип RevocationInfoChoices . . . . .	29
14.4 Тип AlgorithmIdentifier . . . . .	29
14.5 Тип CertificateChoices . . . . .	30
14.6 Тип CertificateSet . . . . .	31
14.7 Тип OriginatorInfo . . . . .	31
14.8 Тип OtherKeyAttribute . . . . .	31
14.9 Тип Attribute . . . . .	31
Приложение А (справочное) Контрольные примеры . . . . .	32
Библиография . . . . .	47

## **Введение**

В настоящих рекомендациях приведено описание форматов кодирования, идентификаторов и форматов параметров при использовании ГОСТ Р 34.10, ГОСТ Р 34.11, ГОСТ Р 34.12, ГОСТ Р 34.13 для криптографической защиты сообщений [сообщений в формате CMS (Cryptographic Message Syntax)].

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

РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ

Информационная технология

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Форматы сообщений, защищенных криптографическими методами

Information technology. Cryptographic information security. Cryptographically secured message formats

Дата введения — 2019—12—01

## 1 Область применения

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

В настоящих рекомендациях дано описание правил использования алгоритмов формирования и проверки подписи в соответствии с ГОСТ Р 34.10, функции хэширования по ГОСТ Р 34.11, функций шифрования в соответствии с ГОСТ Р 34.12 и ГОСТ Р 34.13 для защиты сообщений в формате CMS.

## 2 Нормативные ссылки

В настоящих рекомендациях использованы нормативные ссылки на следующие стандарты и рекомендации:

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

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

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

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

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

ГОСТ Р ИСО/МЭК 8825-1 Информационная технология. Правила кодирования ASN.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

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

Р 1323565.1.012 Информационная технология. Криптографическая защита информации. Принципы разработки и модернизации шифровальных (криптографических) средств защиты информации.

Р 1323565.1.017 Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, сопутствующие применению алгоритмов блочного шифрования

Р 1323565.1.020—2018 Информационная технология. Криптографическая защита информации. Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)

**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:** Стока бит, полученная в результате процесса формирования подписи.

3.1.2 **ключ электронной подписи:** Элемент секретных данных, специфичный для субъекта и используемый только данным субъектом в процессе формирования цифровой подписи.

3.1.3 **ключ проверки электронной подписи:** Элемент данных, математически связанный с ключом подписи и используемый проверяющей стороной в процессе проверки цифровой подписи.

3.1.4 **сертификат ключа проверки электронной подписи:** Электронный документ или документ на бумажном носителе, выданный удостоверяющим центром либо доверенным лицом удостоверяющего центра и подтверждающий принадлежность ключа проверки электронной подписи владельцу сертификата ключа проверки электронной подписи.

3.1.5 **эфемерный ключ:** Элемент секретных данных, генерируемый для каждого сеанса согласования ключей и соответствующий остальным требованиям заданного типа ключа (в том числе уникальности для каждого сообщения или сеанса связи).

**П р и м е ч а н и е —** В некоторых случаях эфемерные ключи используют более одного раза в течение одного сеанса, например для широковещательных приложений.

#### 3.1.6

**хэш-функция:** Функция, отображающая строки бит в строки бит фиксированной длины и удовлетворяющая следующим свойствам:

- 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.  
[ГОСТ Р 34.11—2012, пункт 3.1.6]

#### 3.1.7

**хэш-код:** Стока бит, являющаяся выходным результатом хэш-функции.

[ГОСТ Р 34.11—2012, пункт 3.1.5]

3.1.8 **тэг:** Числовой идентификатор, определяющий тип данных АСН.1 структуры.

#### 3.2 Обозначения

В настоящих рекомендациях использованы следующие обозначения:

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

-  $V_s$  — множество всех двоичных строк длины  $s$ ,  $s \geq 0$ . При  $s = 0$  множество  $V_s$  состоит из единственной пустой строки длины 0;

- $B_s$  — множество байтовых строк длины  $s$ ,  $s \geq 0$ . Стока  $b = (b_1, \dots, b_s)$  принадлежит множеству  $B_s$ , если  $b_1, \dots, b_s \in \{0, \dots, 255\}$ . При  $n = 0$  множество  $B_s$  состоит из единственной пустой строки длины 0;
- $b[i..j] = (b_i, b_{i+1}, \dots, b_j) \in B_{j-i+1}$ , где  $1 \leq i \leq j \leq s$  и  $b = (b_1, \dots, b_s) \in B_s$ ;
- $|A|$  — 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  $B$ ;
- $K_s$  — 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

В настоящих рекомендациях определена структура защищенных данных ContentInfo. Структура ContentInfo инкапсулирует один тип содержимого, данный тип также может быть подвергнут дальнейшей инкапсуляции. В настоящих рекомендациях рассмотрено шесть 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 настоящих рекомендациях рассматриваются варианты использования стандартов на алгоритмы и механизмы, 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 настоящих рекомендациях 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 ASN.1 представлен следующим образом:

```
ContentInfo ::= SEQUENCE
{
    contentType          Content-Type,
    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 настоящих рекомендациях рассмотрены 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» определено 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 байтов, например текстовые файлы ASCII. Такие строки не должны иметь никакой значимой для CMS формата внутренней структуры (несмотря на то что 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 друг от друга.

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 вычисляют значение хэш-функции от содержимого;
- если в сообщении присутствуют подписываемые атрибуты, то значение хэш-функции от содержимого оформляют как еще один подписываемый атрибут (`message-digest`), добавляют в раздел с подписываемыми атрибутами. Далее на весь набор подписываемых атрибутов вычисляют значение хэш-функции и принимают за `hash`. Если в сообщении подписываемые атрибуты отсутствуют, то значение хэш-функции от содержимого принимают за `hash` (см. 7.4);
- для каждого отправителя полученное значение хэш-функции `hash` подписывают с использованием его ключа ЭП;
- для каждого отправителя значение подписи и другую специфичную для данного отправителя информацию записывают в структуру `SignerInfo` (см. 7.3). На этом шаге также записывают сертификаты и САС как для отправителя, осуществлявшего подпись, так и для остальных отправителей;
- значения хэш-функций для всех отправителей и структуру `SignerInfo` для всех отправителей записывают вместе с содержимым в структуру `SignedData` (см. 7.1).

Получатели, осуществляющие проверку подписи, независимо (самостоятельно) вычисляют значение хэш-функции. Вычисленное значение и ключ проверки ЭП подписавшего отправителя используют 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;
- посредством идентификатора ключа субъекта (`Subject Key Identifier`), который однозначно идентифицирует ключ проверки ЭП отправителя.

Сертификат отправителя, осуществлявшего подпись, может быть включен в структуру `SignedData`. Данное включение не является обязательным.

Если в сообщении присутствует несколько подписей, то успешную проверку одной подписи, связанной с данным подписавшим, как правило, трактуют как успешную проверку подписи отправителя.

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. Это позволяет получателям проверять подписи, полученные с помощью разных алгоритмов.

## 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» представлена в виде структуры SignedData. Структура SignedData в формате ASN.1 приведена следующим образом:

```
SignedData ::= SEQUENCE
{
    version CMSVersion,
    digestAlgorithms DigestAlgorithmIdentifiers,
    encapsContentInfo 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 — в 7.2, тип SignerInfo — в 7.3.

Поля структуры SignedData имеют следующий смысл:

- **version** — версия синтаксиса. Точное значение зависит от полей CertificateSet, eContentType и SignerInfo. Версия синтаксиса должна быть определена согласно [1], 5.1;

- **digestAlgorithms** — набор идентификаторов алгоритмов хэширования. Набор может состоять из любого числа элементов, не 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, можно не проверять. Процесс вычисления значения хеш-функции описан в 7.4;

- **encapContentInfo** — подписываемое содержимое; которое состоит из идентификатора типа содержимого и самого содержимого. Подробнее структура EncapsulatedContentInfo описана в 7.2;

- **certificates** — набор сертификатов. Предполагается, что множество сертификатов достаточно, для того 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 САС достаточно, для того 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. 7.3). Так как каждый отправитель может использовать различные методы ЭП и будущие спецификации могут обновить синтаксис, все реализации должны корректно обрабатывать неподдерживаемые версии SignerInfo. Все реализации должны корректно обрабатывать неподдерживаемые алгоритмы подписи в случае их использования.

## 7.2 Тип EncapsulatedContentInfo

Тип `EncapsulatedContentInfo` в формате АСН.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 байтов. Содержимое не обязано кодироваться в DER.

Опциональный пропуск параметра `eContent` позволяет создавать так называемые «отделяемые подписи». В случае внешней подписи подписьляемое 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, в котором отсутствуют субъект подписи, использование значения `EncapsulatedContentInfo` нецелесообразно. В этом случае тип подписываемого содержимого в `EncapsulatedContentInfo` должен представлять собой «простые данные» (см. раздел 6), а содержимое поля `EncapsulatedContentInfo` должно быть опущено.

## 7.3 Тип SignerInfo

Информация для каждого подписавшего представлена 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 быть определена согласно [1], 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) 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 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 быть использован для идентификации ключа проверки ЭП, который не соответствует формату сертификатов X.509;

- `digestAlgorithm` — определение алгоритма вычисления хэш-функции. Значение хэш-функции вычисляется либо от содержимого, либо от содержимого вместе с подписанными атрибутами; этот процесс описан в 7.4. Идентификатор алгоритма хэширования должен присутствовать в числе тех, которые перечислены в поле `digestAlgorithms` структуры `SignedData`;

- `signedAttrs` — набор подписанных атрибутов для подписи. Поле не является обязательным, но оно должно присутствовать, если тип содержимого структуры `EncapsulatedContentInfo` представляет собой не «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. 13.1). Атрибут `content-type` не должен быть использован в не подписываемых атрибутах удостоверяющих подписей (см. 13.3),

- `message-digest` — атрибут, содержащий значение хэш-функции от содержимого (см. 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. 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 определен 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 байтов, к которому применяется процесс вычисления подписи. Хэшируется только содержимое строки байтов без хэширования байтов тэга 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 байты, которые составляют значение `SignedData.encapContentInfo.eContent` (например, содержимое файла) являются входом для вычисления значения хэш-функции. Преимущество этого варианта заключается 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 `signedAttrs`, 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 13.3. Кодирование поля `signedAttrs` выполняется отдельно для вычисления значения хэш-функции. Для DER-кодирования в `signedAttrs` вместо тэга `IMPLICIT [0]` использован тэг `EXPLICIT SET OF`, который должен быть включен в процесс вычисления значения хэш-функции вместе с длиной и строкой байтов поля `SignedAttributes`.

Несмотря на то что байты тэга и длины `SignedData.encapContentInfo.eContent` не включены в процесс получения значения хэш-функции, они защищены другими средствами. Целостность значения поля длины обусловлено криптографическими свойствами функции хэширования, так как алгоритмически сложно найти два сообщения различной 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.

Значение хэш-функции для последующего использования в алгоритме подписи необходимо вычислять согласно 7.4.

Детали процесса формирования подписи зависят от используемого алгоритма подписи (см. 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.

Значение хэш-функции для последующей проверки необходимо вычислять согласно 7.4. Если в сообщении присутствуют подписываемые атрибуты, то вычисленное значение хэш-функции необходимо сравнивать с представленным в сообщении значением атрибута `messageDigest`, включенного в подписанные атрибуты `SignedData.signerInfo`. В случае несовпадения следует считать проверку сообщения недействительной.

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

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 могут быть основаны на построении и проверке цепочки сертификатов, а также могут зависеть от других внешних условий. Детали проверки подписи связаны с используемым алгоритмом (см. 7.7).

## 7.7 Специфика данных типа «подписанные данные»

В данном подразделе описано использование алгоритмов подписи по ГОСТ Р 34.10 в CMS.

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

Значения подписи указаны в поле `signature` структуры `SignerInfo`, вложенной в структуру `SignedData`, а также в поле подписи `SignerInfo` атрибутов удостоверяющей подписи.

Идентификаторы объектов АСН.1, параметры, битовое представление в точности совпадает с представлением подписи в сертификате, описанном в [2].

Сертификат, используемый для проверки подписи сообщений, должен содержать расширение `KeyUsage`, а данное расширение, в свою очередь, — флаг `digitalSignature`.

### 7.7.1 Специфика использования ГОСТ Р 34.10 с ключом 256 бит

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

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

### 7.7.2 Специфика использования ГОСТ Р 34.10 с ключом 512 бит

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

лении big-endian (старший байт записывается первым), а вторые 64 байта содержат число  $r$  в представлении big-endian.

```
GostR3410-2012-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 получателя или более. Комбинация зашифрованного содержимого и одного зашифрованного ключа для одного получателя называется цифровым конвертом получателя. Любое содержимое может быть упаковано в «конверт данных» для произвольного числа получателей при использовании любого поддерживаемого получателем механизма управления ключами (см. 8.2).

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

- генерируется случайный ключ шифрования содержимого (см. раздел 12);
- ключ шифрования содержимого зашифровывают для каждого получателя. Детали шифрования зависят от используемого механизма управления ключами (см. 8.4). В сообщениях формата CMS с использованием российских криптографических алгоритмов применяют два способа управления ключами:
  - использование симметричного общего ключа, выработанного при помощи ключа проверки ЭП получателя и ключа ЭП отправителя, для шифрования ключа шифрования содержимого (см. 8.2.1, 8.2.2),
  - применение ранее выработанного симметричного ключа для шифрования ключа шифрования содержимого (см. 8.2.3);
- для каждого получателя зашифрованный ключ шифрования содержимого и другая информация, специфичная для получателя, записывают в структуру `RecipientInfo` (8.2);
  - содержимое зашифровывают с помощью ключа шифрования содержимого. При шифровании может возникнуть необходимость дополнения данных содержимого до полного блока (см. 8.3);
  - значение `RecipientInfo` вместе с зашифрованным содержимым для всех получателей записывают в структуру `EnvelopedData` (см. 8.1).

Получатель «распаковывает» содержимое следующим образом: сначала расшифровывает один из зашифрованных ключей шифрования содержимого, затем с помощью полученного ключа — 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 ACH.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 — в 8.2.

Поля структуры EnvelopedData имеют следующий смысл:

- version — номер версии синтаксиса. Версия синтаксиса должна быть определена согласно [1], 6.1;

- originatorInfo — поле, содержащее информацию об отправителе; данное поле присутствует в структуре, если этого требует алгоритм согласования ключей: в случае использования алгоритма согласования ключей на основе статического протокола Диффи—Хеллмана данное поле обязательно должно присутствовать в структуре. В этом случае поле содержит информацию о сертификате, ключ которого использован для согласования ключей. Поле может содержать информацию о нескольких сертификатах и САС. 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 8.3; для всех получателей использован один и тот же алгоритм зашифрования содержимого;

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

Поля структуры ContentEncryptionAlgorithmIdentifier имеют следующий смысл:

- encryptionAlgorithmOID — идентификатор алгоритма шифрования (см. 8.3.1);
- parameters — дополнительные параметры алгоритма шифрования.

Поля структуры Gost3412-15-Encryption-Parameters имеют следующий смысл:

- ukm — ключевой материал пользователя UKM; параметр является обязательным и должен содержать n байт. В зависимости от алгоритма n принимает следующие значения (см. 8.3.1) для алгоритмов на основе блочного шифра:

- ГОСТ Р 34.12 «Магма» — n = 12;
- ГОСТ Р 34.12 «Кузнецик» — n = 16.

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 получателю или более.

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 (сокращения от KeyTransRecipientInfo, KeyAgreeRecipientInfo, KEKRecipientInfo). 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 ACH.1 представлен следующим образом:

```
RecipientInfo ::= CHOICE
```

```
{
```

```
    ktri      KeyTransRecipientInfo,
    kari[1]   KeyAgreeRecipientInfo,
    kekri[2]  KEKRecipientInfo,
    pwri[3]   PasswordRecipientInfo,
    ori[4]    OtherRecipientInfo
}
```

Типы KeyTransRecipientInfo, KeyAgreeRecipientInfo, KEKRecipientInfo определены в 8.2.1—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 эфемерного протокола Диффи—Хеллмана;
- 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 статического протокола Диффи—Хеллмана;
- 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 эфемерного протокола Диффи—Хеллмана, хранится в структуре KeyTransRecipientInfo. Каждый экземпляр структуры KeyTransRecipientInfo имеет ключ шифрования содержимого, зашифрованный для одного из получателей.

Тип KeyTransRecipientInfo в формате ACH.1 представлен следующим образом:

```
KeyTransRecipientInfo ::= SEQUENCE
```

```
{
```

```
    version          CMSVersion,
    rid              RecipientIdentifier,
    keyEncryptionAlgorithm
KeyEncryptionAlgorithmIdentifier,
    encryptedKey     OCTET STRING
}
```

```
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 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 использован отправителем для защиты ключа шифрования содержимого. Сертификат ключа получателя должен содержать в расширении KeyUsage установленный флаг keyAgreement. 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. Значение issuerAndSerialNumber идентифицирует получателя по выделенному имени издателя и порядковому номеру сертификата. Для ссылки на сертификат стандарта X.509 может 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 8.4;

- encryptedKey — поле представляет собой структуру GostR3410-KeyTransport, закодированную в OCTET STRING.

Поля структуры GostR3410-KeyTransport имеют следующий смысл:

- encryptedKey — зашифрованные по алгоритму KExp15 (см. Р 1323565.1.017) ключ шифрования и имитовставка (см. 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт — имитовставку. В зависимости от алгоритма n принимает следующие значения (см. 8.4.1) для алгоритма:

- id-gostr3412-2015-magma-wrap-kexp15 n = 8;
- id-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 байта.

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 каждом пользователе, использующем механизм управления ключами на основе статического и эфемерного протоколов Диффи—Хеллмана, хранится в структуре KeyAgreeRecipientInfo. Каждый экземпляр KeyAgreeRecipientInfo имеет ключ шифрования содержимого для одного или нескольких получателей, которые используют тот же алгоритм соглашения и его одинаковые параметры.

```

KeyAgreeRecipientInfo ::= SEQUENCE
{
    version                  CMSVersion,
    originator[0]            EXPLICIT
OriginatorIdentifierOrKey,
    ukm[1]                   EXPLICIT UserKeyingMaterial,
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     GostR3410-12-KEG-Parameters
}
GostR3410-12-KEG-Parameters ::= 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, который 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 использован 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* — ключевой материал пользователя *UKM*, данный параметр является обязательным и должен содержать 32 байта;
- *keyEncryptionAlgorithm* — определение алгоритма шифрования ключа шифрования содержимого и его параметров; процесс шифрования описан, и значения параметров определены в 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. Р 1323565.1.017) ключ шифрования и имитовставка (см. 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт — имитовставку. В зависимости от алгоритма n принимает следующие значения (см. 8.4.1) для алгоритма:
  - id-gostr3412-2015-magma-wrap-kexp15 n = 8,
  - id-gostr3412-2015-kuznyechik-wrap-kexp15 n = 16.

Сертификат ключа получателя должен содержать 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 применен 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* 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 имеет ключ шифрования содержимого для одного или нескольких получателей, которым ранее распределены ключи шифрования 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 быть всегда 4;
- **kekid** — идентификатор симметричного ключа, который ранее 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 8.4;
- **encryptedKey** — зашифрованные по алгоритму KExp15 (см. Р 1323565.1.017) ключ шифрования и имитовставка (см. 8.4.2): первые 32 байта представляют ключ, оставшиеся n байт — имитовставку. В зависимости от алгоритма n принимает следующие значения (см. 8.4.1) для алгоритма:
  - id-gostr3412-2015-magma-wrap-kexp15 n = 8,
  - id-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 (см. раздел 12). При использовании описанных ниже российских криптографических алгоритмов процедура дополнения содержимого не производится. Операция зашифрования отображает произвольную строку байтов (данные) в другую строку байтов (зашифрованный текст) с использованием ключа шифрования. Зашифрованные данные включают в поле структуры EnvelopedData: OCTET STRING EnvelopedData.encryptedContentInfo.encryptedContent.

#### 8.3.1 Параметры шифрования сообщений

В данном пункте изложены рекомендации, используемые при реализации CMS с поддержкой шифрования содержимого согласно ГОСТ Р 34.12, ГОСТ Р 34.13 и Р 1323565.1.017.

Идентификаторы алгоритма шифрования содержимого указываются в следующих полях:

- EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm;
- EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm (определение структуры EncryptedData см. в разделе 10).

Для шифрования содержимого введены следующие новые 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 произведено с помощью блочного шифра по ГОСТ Р 34.12 «Магма» в режиме CTR-ACPKM согласно Р 1323565.1.017. При этом длина блока гаммы  $s$ , приведенная в Р 1323565.1.017, равна 64 бита; размер секции (см. Р 1323565.1.017) — 8 Кбайт.

```
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 осуществлено с помощью шифра «Магма» по ГОСТ Р 34.12 в режиме выработки имитовставки согласно ГОСТ Р 34.13 (размер имитовставки — 64 бита), а шифрование произведено с помощью шифра «Магма» по ГОСТ Р 34.12 в режиме CTR-ACPKM, определенном в Р 1323565.1.017. При этом длина блока гаммы  $s$ , приведенная в Р 1323565.1.017, равна 64 бита; размер секции (см. Р 1323565.1.017) — 8 Кбайт.

```
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 произведено с помощью блочного шифра «Кузнечик» по ГОСТ Р 34.12 в режиме CTR-ACPKM согласно Р 1323565.1.017. При этом длина блока гаммы  $s$ , приведенная в Р 1323565.1.017, равна 128 бит; размер секции (см. Р 1323565.1.017) — 256 Кбайт.

```
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 «Кузнечик» по ГОСТ Р 34.12 в режиме выработки имитовставки согласно ГОСТ Р 34.13 (размер имитовставки — 128 бит), а шифрование производится с помощью шифра «Кузнечик» по ГОСТ Р 34.12 в режиме CTR-ACPKM, определенном в Р 1323565.1.017. При этом длина блока гаммы  $s$ , приведенная в Р 1323565.1.017, равна 128 бит; размер секции (см. Р 1323565.1.017) — 256 Кбайт.

Алгоритмы шифрования используют для шифрования содержимого, указанного в полях:

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

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

### 8.3.2 Процесс формирования и разбора содержимого

Зашифрование содержимого для алгоритмов `id-gostr3412-2015-magma-ctracpkm` и `id-gostr3412-2015-kuznyechik-ctracpkm` осуществлено следующим способом:

- с использованием ключа шифрования содержимого  $K_{me}$  в зависимости от заданного алгоритма шифрования происходит зашифрование содержимого с начальным значением  $/V$ , равным соответствующим полям ( $n$  — размер поля `ukm`):

- `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`,

- `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`;

- ключ шифрования содержимого  $K_{me}$  зашифрован в соответствии с выбранным механизмом управления ключами и установлен в соответствующее поле.

Расшифрование содержимого для алгоритмов `id-gostr3412-2015-magma-ctracpkm` и `id-gostr3412-2015-kuznyechik-ctracpkm` осуществлено следующим способом:

- извлечен и расшифрован ключ шифрования содержимого  $K_{me}$  (см. 8.4.2);

- с использованием ключа  $K_{me}$  осуществлено расшифрование содержимого с начальным значением  $IV$ , равным соответствующим полям ( $n$  — размер поля `ukm`):

- `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`,

- `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`.

Зашифрование содержимого для алгоритмов `id-gostr3412-2015-magma-ctracpkm-omac` и `id-gostr3412-2015-kuznyechik-ctracpkm-omac` осуществлено следующим способом:

- из ключа  $K_{me}$  с использованием алгоритма `KDF_TREE_GOSTR3411_2012_256`, описанного в Р 50.1.113, вырабатываются два ключа:

- ключ шифрования сообщения  $K(1)$ ,

- ключ имитозащиты  $K(2)$ .

Входные параметры для алгоритма `KDF_TREE_GOSTR3411_2012_256` принимают следующие значения:

$$K_{in} = K_{me}$$

*label* = «*kdf tree*»

*seed* = `ukm[n - 7..n]`

$$R = 1;$$

- с использованием ключа имитозащиты  $K(2)$ , выработанного из  $K_{me}$ , осуществлено вычисление имитовставки для открытого текста содержимого;

- результирующее значение имитовставки добавлено в конец содержимого. Полученная строка байт зашифрован с использованием ключа шифрования  $K(1)$ , выработанного из  $K_{me}$ , с начальным значением  $IV$ , равным соответствующим полям:

- `EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`,

- `EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.parameters.ukm[1..n - 8]`;

- последние  $n$  байт зашифрованного текста, где  $n$  зависит от выбранного алгоритма выработки имитовставки, являющиеся зашифрованной имитовставкой, в виде атрибута `content-mac` помещаются в поле незашифрованных атрибутов сообщения `unprotectedAttrs` (см. 13.4);

- ключ шифрования содержимого  $K_{me}$  зашифрован в соответствии с выбранным механизмом управления ключами и установлен в соответствующее поле.

Расшифрование содержимого для алгоритмов `id-gostr3412-2015-magma-ctracpkm-omac` и `id-gostr3412-2015-kuznyechik-ctracpkm-omac` осуществлено следующим способом:

- извлечен и расшифрован ключ шифрования содержимого  $K_{me}$  (см. 8.4.2);

- из ключа  $K_{me}$  с использованием алгоритма `KDF_TREE_GOSTR3411_2012_256`, описанного в Р 50.1.113, вырабатываются два ключа:

- ключ шифрования сообщения  $K(1)$ ,

- ключ имитозащиты  $K(2)$ .

Входные параметры для алгоритма `KDF_TREE_GOSTR3411_2012_256` принимают следующие значения:

$$K_{in} = K_{me}$$

*label* = «*kdf tree*»

*seed* = `ukm[n - 7..n]`

$$R = 1.$$

Зашифрованная имитовставка из атрибута `content-mac` (см. 13.4) помещена в конец зашифрованного содержимого;

- с использованием ключа шифрования  $K(1)$ , выработанного из  $K_{me}$ , осуществлено расшифрование с начальным значением IV, равным соответствующим полям:

- EnvelopedData.EncryptedContentInfo.contentEncryptionAlgorithm.

parameters.ukm[1..n - 8],

- EncryptedData.EncryptedContentInfo.contentEncryptionAlgorithm.

parameters.ukm[1..n - 8];

- последние  $n$  байт расшифрованного текста, где  $n$  зависит от выбранного алгоритма выработки имитовставки, обозначены как mac-text;

- с использованием ключа имитозащиты  $K(2)$ , выработанного из  $K_{me}$ , осуществлено вычисление имитовставки для расшифрованного текста за исключением имитовставки. Результатирующее значение обозначено как mac-count;

- 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 осуществлено 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 байтов (данные) в другую строку байтов (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 пункте изложены рекомендации, используемые при реализации CMS с поддержкой шифрования ключей согласно алгоритмам KExp15 (см. Р 1323565.1.017).

Алгоритм зашифрования ключа описан 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. Поле algorithm данной структуры принимает 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) выбирают KEG-256 или KEG-512.

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

- в зависимости от механизма управления ключами вырабатываются ключи  $KEK_t$  и  $KIM_t$ ;
- в зависимости от алгоритма зашифрования ключа происходит зашифрование ключа на ключе  $KEK_t$  с выработкой имитовставки на ключе  $KIM_t$  по алгоритму KExp15 (см. Р 1323565.1.017).

Процедуру расшифрования выполняют аналогично.

#### 8.4.2 Процесс формирования и разбора ключей

##### 8.4.2.1 Процесс формирования и разбора ключей на основе статического протокола Диффи—Хеллмана

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

- проверяют соответствие параметров ключа проверки ЭП отправителя и ключа проверки ЭП получателя. При несовпадении параметров алгоритм завершается с ошибкой;

- в качестве ключа  $K_s$  принимают ключ ЭП отправителя;

- заполняют поле `EnvelopedData.recipientInfos.kari.originator` в зависимости от желаемого способа задания ключа проверки ЭП отправителя (см. 8.2.2). Параметры и битовое представление ключа проверки ЭП в точности совпадают с их представлением в сертификате (см. [2]);

- вырабатывается уникальный (случайный) ненулевой  $UKM$  длиной 32 байта, который кодируется по правилам типа OCTET STRING и записывается в поле `EnvelopedData.recipientInfos.kari.ukm`;

- для каждого получателя вырабатываются два ключа  $KEK_t$  и  $KIM_t$  с использованием ключа  $K_s$ , ключа проверки ЭП получателя  $P_r$ ,  $UKM$  и алгоритма KEG (см. Р 1323565.1.020—2018, 6.4.5):

$$KIM_t \parallel KEK_t = \text{KEG}(K_s, P_r, UKM[1..24]);$$

- вырабатывается случайный симметричный ключ  $K_m$ , соответствующий алгоритму, указанному в поле `EnvelopedData.encryptedContentInfo.contentEncryptionAlgorithm`;

- для каждого получателя осуществлен экспорт ключа  $K_m$  на ключах  $KEK_t$  и  $KIM_t$  с использованием алгоритма экспорта KExp15 (см. Р 1323565.1.017), при этом в качестве ключа  $K_{MAC}^{EXP}$  принимается ключ  $KIM_t$ ; в качестве ключа  $K_{ENC}^{EXP}$  — ключ  $KEK_t$ . В качестве IV используется  $UKM[25..24 + n/2]$ , где  $n$  — длина блока, выраженная в байтах;

- результатом работы алгоритма KExp15 является строка KEXP, содержащая в себе зашифрованный ключ и имитовставку в соответствии с Р 1323565.1.017;

- результирующая строка KEXP записана в поле `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey`.

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

- извлекают строку KEXP из поля `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey`, содержащую в себе зашифрованный ключ  $K_m^{EXP}$  и имитовставку согласно Р 1323565.1.017;

- выбирают алгоритм и параметры ключа проверки ЭП отправителя  $P_s$  из структуры `EnvelopedData.recipientInfos.kari.originator` с помощью либо сертификата, либо ключа проверки ЭП. Проверяют соответствие алгоритма и параметров собственного ключа ЭП параметрам ключа проверки ЭП отправителя  $P_s$ . При несоответствии параметров алгоритм завершается с ошибкой;

- вырабатываются два ключа  $KEK_t$  и  $KIM_t$  с использованием ключа ЭП получателя  $K_r$ , ключа проверки ЭП отправителя  $P_s$ ,  $UKM$  и алгоритма KEG (см. Р 1323565.1.020—2018, 6.4.5.1):

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

- осуществлены импорт и проверка имитозащиты ключа  $K_m^{EXP}$  на ключах  $KEK_t$  и  $KIM_t$  с использованием алгоритма Klmp15 (см. Р 1323565.1.017). При этом в качестве ключа  $K_{MAC}^{EXP}$  принимается ключ  $KIM_t$ ; в качестве ключа  $K_{ENC}^{EXP}$  — ключ  $KEK_t$ . Результатом импорта является ключ  $K_m$ .

8.4.2.2 Процесс формирования и разбора ключей на основе эфемерного протокола Диффи—Хеллмана

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

- проверяют, содержат или не содержат все сертификаты получателей расширения KeyUsage и присутствие флага keyAgreement. В последнем случае должен отсутствовать флаг encipherOnly;

- генерируется случайный эфемерный ключ  $K_E$  с использованием алгоритма и параметров ключа получателя (см. 12). Обозначают точку  $P_E$  на эллиптической кривой  $E$ , вычисляемую по следующему равенству:

$$P_E = [K_E]P = \underbrace{P + \dots + P}_{k \text{ раз}}$$

где  $P$  — фиксированная точка эллиптической кривой  $E$ , определяемая значением идентификатора, содержащегося в сертификате ключа проверки ЭП  $P_r$ ;

- эфемерный случайный ключ может быть создан как отдельно для каждого получателя, так и один для любого числа получателей;

- заполняются поля `EnvelopedData.recipientInfos.ktri.rid` или `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.rid` в зависимости от выбранного алгоритма согласования ключей. Алгоритм и параметры эфемерного ключа проверки ЭП отправителя помещаются в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.ephemeralPublicKey` или в поле `EnvelopedData.recipientInfos.kari.originator.originatorKey.algorithm`. Ключ проверки ЭП  $P_E$  помещается в поле `ephemeralPublicKey.publicKey` или в поле `EnvelopedData.recipientInfos.kari.originator.originatorKey.publicKey`. Идентификаторы объектов АСН.1, параметры, битовое представление в точности совпадает с представлением подписи в сертификате, описанным в [2];

- вырабатывается уникальный (случайный) ненулевой *UKM* длиной 32 байта, который кодируется по правилам типа OCTET STRING и записывается в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.ukm` или в поле `EnvelopedData.recipientInfos.kari.ukm`;

- вырабатываются два ключа  $KEK_t$  и  $KIM_t$  с использованием ключа  $K_E$ , ключа проверки ЭП получателя  $P_r$ , *UKM* и алгоритма KEG (см. Р 1323565.1.020—2018, 6.4.5.1):

$$KIM_t \parallel KEK_t = KEG(K_E, P_r, UKM);$$

- вырабатывается случайный симметричный ключ  $K_m$ , соответствующий алгоритму в поле `EnvelopedData.encryptedContentInfo.contentEncryptionAlgorithm`;

- осуществляется экспорт ключа  $K_m$  на ключах  $KEK_t$  и  $KIM_t$  с использованием алгоритма экспорта KExp15 (см. Р 1323565.1.017), при этом в качестве ключа  $K_{MAC}^{EXP}$  принимается ключ  $KIM_t$ ; в качестве ключа  $K_{ENC}^{EXP}$  — ключ  $KEK_t$ . В качестве IV использован *UKM*  $[25..24 + n/2]$ , где  $n$  — длина блока, выраженная в байтах;

- результатом работы алгоритма KExp15 является строка KEXP, содержащая в себе зашифрованный ключ и имитовставку в соответствии с Р 1323565.1.017;

- результирующая строка KEXP записана в поле `EnvelopedData.recipientInfos.ktri.encryptedKey.encryptedKey` или в поле `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey`.

Если сообщение отправлено нескольким получателям с различными параметрами ключа проверки ЭП, то для каждого набора параметров генерируется свой эфемерный ключ (см. 12) и создается своя структура `KeyTransRecipientInfo` или `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  $K_r$  применен следующий алгоритм:

- извлекают строку KEXP из поля `EnvelopedData.recipientInfos.ktri.encryptedKey.encryptedKey` или из поля `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.encryptedKey`, содержащую зашифрованный ключ  $K_m^{EXP}$  и имитовставку согласно Р 1323565.1.017;

- выбирают алгоритм, параметры и ключ проверки ЭП отправителя  $P_E$  из структуры `EnvelopedData.recipientInfos.ktri.rid` или `EnvelopedData.recipientInfos.kari.recipientEncryptedKeys.rid` с помощью либо сертификата, либо ключа проверки ЭП. Проверяют соответствие алгоритма и параметров собственного ключа ЭП параметрам ключа проверки ЭП отправителя  $P_E$ . При несоответствии параметров алгоритм завершается с ошибкой;

- вырабатываются два ключа  $KEK_t$  и  $KIM_t$  с использованием ключа ЭП получателя  $K_r$ , ключа проверки ЭП отправителя  $P_E$ , *UKM* и алгоритма KEG (см. Р 1323565.1.020—2018, 6.4.5.1):

$$KIM_t \parallel KEK_t = \text{KEG}(K_r, P_E, UKM);$$

- осуществляются импорт и проверка имитозащиты ключа  $K_m^{\text{EXP}}$  на ключах  $KEK_t$  и  $KIM_t$  с использованием алгоритма KImp15 (см. Р 1323565.1.017). При этом в качестве ключа  $K_{\text{MAC}}^{\text{EXP}}$  принимается ключ  $KIM_t$ ; в качестве ключа  $K_{\text{ENC}}^{\text{EXP}}$  — ключ  $KEK_t$ . Результатом импорта является ключ  $K_m$ .

## 9 Содержимое типа «хэшированные данные»

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

Как правило, тип содержимого «хэшированные данные» используют для контроля целостности содержимого, а результат становится входом для содержимого типа «конверт данных».

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

- определение алгоритма хэширования и вычисление значения хэш-функции от содержимого;
- запись идентификатора алгоритма вычисления хэш-функции и значения хэш-функции вместе с содержимым в структуру `DigestedData`.

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

### 9.1 Тип `DigestedData`

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» представлено в виде структуры `DigestedData`. Структура `DigestedData` в формате ACH.1 приведена следующим образом:

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

Типы `CMSVersion` и `AlgorithmIdentifier` определены в разделе 14; тип `EncapsulatedContentInfo` — в 7.2.

Поля типа `DigestedData` имеют следующий смысл:

- `version` — номер версии синтаксиса. Версия синтаксиса должна быть определена согласно [1], раздел 7;
- `digestAlgorithm` — определение алгоритма хэширования и его параметров. Процесс вычисления хэш-функции аналогичен процессу, описанному в 7.4, за исключением того, что отсутствуют подписанные атрибуты. Если поле `parameters` не использовано, его значение должно отсутствовать полностью;

- `encapContentInfo` — хэшируемое содержимое в соответствии с определением согласно 7.2;

- `digest` — результат хэширования.

Порядок следования полей в `digestAlgorithm`, `encapContentInfo`, `digest` позволяет обработать значение `DigestedData` за один проход.

### 9.2 Специфика данных типа «хэшированные данные»

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

Формат и идентификаторы алгоритмов полностью соответствуют [3].

Использование хэш-кодов в разных типах содержимого в форматах:

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

### 9.2.1 Сообщения с длиной хэш-кода 256 бит

Алгоритм хэширования по ГОСТ Р 34.11 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 байта:

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

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

### 9.2.2 Сообщения с длиной хэш-кода 512 бит

Алгоритм хэширования по ГОСТ Р 34.11 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 байта:

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

Представление значения хэш-кода соответствует значению  $h$  алгоритма ГОСТ Р 34.11 в виде строки байтов и записывается в формате little-endian (старший байт 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.

Как правило, тип «зашифрованные данные» применяют для хранения шифрованных локальных данных.

### 10.1 Тип `EncryptedData`

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» приведено в виде структуры `EncryptedData`. Структура `EncryptedData` в формате ASN.1 представлена 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 быть определена согласно [1], раздел 8;
- `encryptedContentInfo` — зашифрованное содержимое (подробнее см. 8.1);
- `unprotectedAttrs` — набор незашифрованных атрибутов, являющийся необязательным полем.

## 10.2 Специфика данных типа «зашифрованные данные»

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

## 11 Содержимое типа «аутентифицированные данные»

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

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

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

- генерируется случайный ключ для конкретного алгоритма вычисления имитовставки (см. 12);
- ключ имитозащиты зашифровывается для каждого получателя. Подробности этого процесса зависят от используемого механизма управления ключами (более подробно см. 8.4);
- для каждого получателя зашифрованный ключ имитозащиты и другая информация, специфичная для данного получателя, записываются в структуру `RecipientInfo`, определенную в 8.2;
- используя ключ имитозащиты, отправитель вычисляет имитовставку для содержимого. Если отправитель использует дополнительное содержимое для аутентификации (см. 11.2), то вычисляется хэш-код от содержимого, а от полученного результата и другой информации — имитовставка.

### 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» представлено в виде структуры AuthenticatedData. Структура AuthenticatedData в формате ASN.1 приведена следующим образом:

```
AuthenticatedData ::= SEQUENCE  
{  
    version          CMSVersion,  
    originatorInfo[0] IMPLICIT OriginatorInfo OPTIONAL,  
    recipientInfos   RecipientInfos,  
    macAlgorithm     MessageAuthenticationCodeAlgorithm,  
    digestAlgorithm[1] DigestAlgorithmIdentifier OPTIONAL,  
    encapsContentInfo 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 — в 8.2, тип EncapsulatedContentInfo — в 7.2.

Поля структуры AuthenticatedData имеют следующий смысл:

- **version** — номер версии синтаксиса. Версия синтаксиса должна быть определена согласно [1], 9.1;

- **originatorInfo** — информация об отправителе, необязательное поле, отсутствует для российских криптографических алгоритмов;

- **recipientInfos** — список информации о каждом из получателей, данный список не может быть пустым (подробнее см. 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 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 согласно 7.2;

- **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. 13.1);
- `message-digest` — атрибут, содержащий значение хэш-функции от содержимого (см. 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:

- при отсутствии атрибутов в поле `authAttrs` имитовставка вычисляется для содержимого поля `encapContentInfo.eContent`;
- при наличии атрибутов в поле `authAttrs` 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 байтов. Входными параметрами для алгоритма вычисления имитовставки являются только те байты, которые соответствуют значению `eContent`; байты тэга и длины не следует использовать. 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. 13.1) и атрибут `message-digest` (см. 13.2) должны быть включены в результат вычисления имитовставки, и входным параметром для алгоритма вычисления имитовставки является закодированный атрибут `authAttrs` в DER-кодировке. Тэг `IMPLICIT [2]` в поле `authAttrs` не используется для DER-кодирования, для этого применяется тэг `EXPLICIT SET OF`. То есть в вычисление имитовставки должен быть включен DER закодированный тэг `SET OF`, а не тэг `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 байты, которые содержат значение `encapContentInfo.eContent`, служат входом для алгоритма вычисления хэш-функции без включения байтов тэга и длины. Это становится преимуществом, так как длина аутентифицированного содержимого не должна быть известна заранее. Несмотря на то что байты тэга и длины поля `encapContentInfo.eContent` не включены в процесс вычисления хэш-функции, они по-прежнему защищены, но другим способом. Целостность значения поля длины обусловлена криптографическими свойствами функции хэширования, так как алгоритмически сложно найти два сообщения различной 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, передается в поле `macAlgorithm`. Имитовставка, вычисленная отправителем, кодируется в строку байтов 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. 11.2) и ключ аутентификации, передаваемый в структуре `recipientInfo`. Подробности вычисления имитовставки зависят от используемого алгоритма (см. 11.4).

Получатель не должен полагаться на вычисленные отправителем значения имитовставки и хэш-функции. Содержимое аутентифицируется в соответствии с описанием, приведенным в 11.2. Если отправитель включил аутентифицируемые атрибуты, то содержимое поля `authAttrs` определяется в соответствии с 11.2. Получатель должен вычислить значение хэш-функции от сообщения и убедиться в его совпадении со значением `message-digest` в составе `authAttrs`. Для успешной аутентифика-

ции значение имитовставки, вычисленной получателем, должно совпадать со значением имитовставки в поле `mac`, также должны совпадать соответствующие значения хэш-функций.

#### 11.4 Специфика данных типа «аутентифицированные данные»

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

Идентификаторы алгоритма указаны в поле `AuthenticatedData.macAlgorithm`.

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

Ключ аутентификации генерируется случайным образом (см. 12) и должен быть защищен на основе алгоритма согласования ключей, изложенного в 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 с длиной хэш-кода 256 бит (см. Р 1323565.1.017).

В настоящих рекомендациях 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 с длиной хэш-кода 512 бит (см. Р 1323565.1.017).

В настоящих рекомендациях 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 обеспечена использованием статического протокола Диффи—Хеллмана для содержимого типов «конверт данных» и «аутентифицированные данные». Для обеспечения аутентификации содержимого типа «конверт данных» с использованием эфемерного протокола Диффи—Хеллмана 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 статического протокола Диффи—Хеллмана.

При использовании статического протокола Диффи—Хеллмана не рекомендуется использовать представление ключа проверки ЭП отправителя в виде чистого ключа (`originatorKey` [1] `OriginatorPublicKey` в структуре, приведенной в 8.2.2), так как данный вариант не защищен от подмены отправителя.

Для генерации случайных данных (в том числе и случайных ключей) необходимо использовать генераторы случайных чисел согласно Р 1323565.1.012.

## 13 Полезные атрибуты

В данном разделе определены атрибуты, которые могут быть использованы с содержимым типа

- «подписанные данные»;
- «конверт данных»;
- «хэшированные данные»;
- «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 тогда, 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, неаутентифицированным 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 ACH.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 допускаются.

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 данного типа «аутентифицированные данные» значение хэш-функции — 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 неподписанным атрибутом, неаутентифицированным атрибутом или незащищенным атрибутом.

Следующий идентификатор определяет атрибут `message-digest`:

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

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

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

Атрибут `message-digest` должен содержать единственное значение, несмотря на то что 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 эк-

зимпляя 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 байты подписи для данных типа «подписанные данные». Значение хэш-функции для вычисления удостоверяющей подписи формируется из непосредственно байтов самой подписи, не включая в себя байты тэга и длины. Таким образом, атрибут `countersignature` удостоверяет другую подпись.

Атрибут `countersignature` должен быть неподписаным атрибутом и не должен быть подписаным атрибутом, аутентифицированным атрибутом, неаутентифицированным 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 ACH.1 представлен типом `Countersignature`:

```
Countersignature ::= SignerInfo
```

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

- поле `signedAttributes` не должно содержать атрибут `content-type`; для удостоверяющих подписей содержимое отсутствует;
- поле `signedAttributes` должно содержать атрибут значения хэш-функции, если он содержит другие атрибуты;
- входными параметрами для алгоритма вычисления хэш-функции служат байты поля `signatureValue` структуры `SignerInfo`, представленного в DER-кодировке.

Атрибут `countersignature` может содержать несколько значений. Синтаксис определяет SET OF `AttributeValue` 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 вложенную структуру из атрибутов `countersignature`.

### 13.4 Атрибут `content-mac`

Атрибут `content-mac` 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,  
    encryptedMacs            SET OF EncryptedMac,  
}  
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-attribute(1)  
}
```

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

## 14 Полезные типы

### 14.1 Тип 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 Тип 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 Тип 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 для того, чтобы проверить корректность сертификата на отзыв, но эта проверка не носит обязательный характер. Тип RevocationInfoChoice предоставляет два варианта указания источников информации об аннулированных сертификатах. Первый — crl — списки аннулированных сертификатов, второй — other — любые другие источники информации.

### 14.4 Тип AlgorithmIdentifier

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

Данный тип представляет собой структуру со следующими значениями полей:

- algorithm — идентификатор используемого алгоритма;
- parameters — параметры используемого алгоритма.

#### 14.5 Тип 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 Тип CertificateSet

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

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

#### 14.7 Тип 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, а также атрибутивные сертификаты, ассоциированные с отправителем;
- crls — списки аннулированных сертификатов. Предполагается, что САС достаточны, для того чтобы проверить корректность сертификата на отзыв, но эта проверка не носит обязательный характер.

#### 14.8 Тип 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 Тип 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.

**Приложение А  
(справочное)**

**Контрольные примеры**

В настоящем приложении приведены примеры сообщений в формате CMS при использовании алгоритмов формирования и проверки подписи в соответствии с ГОСТ Р 34.10, функции хэширования по ГОСТ Р 34.11, функций шифрования согласно ГОСТ Р 34.12 и ГОСТ Р 34.13.

Во всех контрольных примерах ниже использованы следующие данные:

А.1 Данные удостоверяющего центра:

- ключ ЭП УЦ;

092F8D059E97E22B90B1AE99F0087FC4D26620B91550CBB437C191005A290810<sub>16</sub>

- идентификатор кривой УЦ;

1.2.643.7.1.2.1.1.1

- сертификат УЦ, представленный в кодировке BASE64:

MII8DCCAZ2gAwIBAgIEAYy6gTAKBggqhQMHAQEDAja4MQ0wCwYDVQQKEwRUSzI2  
MScwJQYDVQQDEX5DQSBUzI2OibHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw  
MTAxMDAwMDAwWhcNNDkxMjMxMDAwWjA4MQ0wCwYDVQQKEwRUSzI2MScwJQYD  
VQQDEX5DQSBUzI2OibHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwaDAhBggqhQMHAQEB  
ATAVBgkqhQMHAQIBAQEGCCqFAwcBAQICA0MABEAaSoKcjw54UACci6svELNF0IYM  
RIW8urUsqamIp0G46XCqrVOuI6Q13N4dwcRsbZdqByf+GC2f5zf03baN5bTKo4GF  
MIGCMGEA1UdAQRaMFIAFIDZDPeZ+GZNk1OjjsCecS2npzESoTowODENMASGA1UE  
ChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsyNjogR09TVCAzNC4xMC0xMiAyNTYtYml0  
ggQBjLqBMB0GA1UdDgQWBBSA2Qz3mfhmTZNTiY7AnnEtp6cxEjAKBggqhQMHAQED  
AgNBAAgV248F4OeNCkh1zJWeC0evHYNb1Szkl1Dm0F875B7CqMrKh2MtJHXenbj  
Gc2uRn2IwgmSf/LZDrYsKKqZSxk=

А.2 Данные отправителя с ключом длины 256 бит:

- ключ ЭП отправителя с ключом длины 256 бит:

0B20810E449978C7C3B76C6FF77A16C532421139344A058EF56310B6B6F377E8<sub>16</sub>

- идентификатор кривой отправителя с ключом длины 256 бит:

1.2.643.7.1.2.1.1.1

- сертификат отправителя с ключом длины 256 бит, представленный в кодировке BASE64:

MII8zCCAxCgAwIBAgIEAYy6gjAKBggqhQMHAQEDAja4MQ0wCwYDVQQKEwRUSzI2  
MScwJQYDVQQDEX5DQSBUzI2OibHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw  
MTAxMDAwMDAwWhcNNDkxMjMxMDAwWjA7MQ0wCwYDVQQKEwRUSzI2MSowKAYD  
VQQDEyFPUKlHSU5BVE9SOiBHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwaDAhBggqhQMh  
AQEBATAVBgkqhQMHAQIBAQEGCCqFAwcBAQICA0MABECWK0TY11qg4GmY3tBJiyz  
pxUN+aOV9WbmTUinqrnEHP7KCNzoAzFg+04SSQpNNShpQnm+jLAZhuJaJfqZ6VbT  
o4GFMIGCMGEA1UdAQRaMFIAFIDZDPeZ+GZNk1OjjsCecS2npzESoTowODENMASG  
A1UEChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsyNjogR09TVCAzNC4xMC0xMiAyNTYt  
Yml0ggQBjLqBMB0GA1UdDgQWBBSRnChHSWbQYwnJC62n2zu5Njd03zAKBggqhQMh  
AQEDAgNBAB41oiijaXSEn58178y2rhxy35/1KEq4XWZ70FtsN1VxWATyzgO5Wliwn  
t1O4GoZsxx8r6T/i7VG65UnmQ1wdOKQ=

А.3 Данные получателя с ключом длины 256 бит:

- ключ ЭП получателя с ключом длины 256 бит:

0DC8DC1FF2BC114BABC3F1CA8C51E4F58610427E197B1C2FBDBA4AE58CBFB7CE<sub>16</sub>

- идентификатор кривой получателя с ключом длины 256 бит:

1.2.643.7.1.2.1.1.1

- сертификат получателя с ключом длины 256 бит, представленный в кодировке BASE64:

MII8jCCAZ+gAwIBAgIEAYy6gzAKBggqhQMHAQEDAja4MQ0wCwYDVQQKEwRUSzI2  
MScwJQYDVQQDEX5DQSBUzI2OibHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw  
MTAxMDAwMDAwWhcNNDkxMjMxMDAwWjA6MQ0wCwYDVQQKEwRUSzI2MSkwJwYD  
VQQDEyBSRUNJUE1FT1Q6IEDPU1QgMzQuMTAtMTIgMjU2LWJpdDBoMCEGCCqFAwcB  
AQEBMBUGCSqFAwcBAgEBAQYIKoUDBwEBAgIDQwAEQL8ngh1zLGMKWHuWhNMPMN5u  
L6SkGqRiJ6qZxZb+4dPKbBT9LNvNKtwUed+BeE5kfqOfolPgFusnL1rnO9yREOj  
gYUwgYIwYQYDVR0BBFowWIAUgNkM95n4Zk2TU4m0wJ5xLaenMRKhOja4MQ0wCwYD

VQQKEwRUSzI2MSCwJQYDVQQDEX5DQSBUSzI2OiBHT1NUIDM0LjEwLTEyIDI1Ni1i  
 aXSCBAGMuoEwHQYDVR0OBByEFLue+Pub90e+pziBU+MvNejjgrzFMAoGCCqFAwcB  
 AQMCA0EAPP9oAd1/5jwokSjPpccsQ0xCdVYM+mGQ0IbpizxQj8gnkt8sq4jR6Ya+  
 I/BDkbZNDNE27TU1p3t5rE9NMEeViA==

#### A.4 Данные отправителя с ключом длины 512 бит:

- ключ ЭП отправителя с ключом длины 512 бит:

F95A5D44C5245F63F2E7DF8E782C1924EADCB8D06C52D91023179786154CBDB156  
 1B4DF759D69F67EE1FBD5B68800E134BAA12818DA4F3AC75B0E5E6F9256911<sub>16</sub>

- идентификатор кривой отправителя с ключом длины 512 бит:

1.2.643.7.1.2.1.2.1

- сертификат отправителя с ключом длины 512 бит, представленный в кодировке BASE64:

MIICNjCCAeOgAwIBAgIEAYy6hDAKBggqhQMHAQEDAja4MQ0wCwYDVQQKEwRUSzI2  
 MSCwJQYDVQQDEX5DQSBUSzI2OiBHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw  
 MTAxMDAwMDAwWhcNNDkxMjMxMDAwMDAwWjA7MQ0wCwYDVQQKEwRUSzI2MSowKAYD  
 VQQDEyFPUK1HSU5BVE9SOiBHT1NUIDM0LjEwLTEyIDUXMi1iaXQwgaowiQYIKoUD  
 BwEBAQIwFQYJKoUDBwECAQIBBggqhQMHAQECAwOBhAAEgYC0i7davCkOGGVcYqFP  
 tS1fUIROzB0fYARIe0tc1TRpare/qzRuVRapqzzO+K21LDpYVfDPs2Sqa13ZN+Ts  
 /JU1v59qCFB2cYpFyB/0kh4+K79yvz7r8+4WE0EmZf8T3ae/J1Jo6xGunech1/G4  
 hMts9HYLnxbwJDMNVGuIHV6gqzOBhTCBgjBhBgNVHQEEWjBYgBSA2Qz3mfhmTZNT  
 iY7AnnEtp6cxEqE6MDgxDTALBgNVBAoTBFRlmjYxJzAlBgNVBAMThkNBIFRLmjY6  
 IEdPU1QgMzQuMTAtMTIgMjU2LWJpdIIIEAYy6gTAdBgNVHQ4EFgQUK+19HAscONGx  
 zCcRpxRAmFHvlXowCgYIKoUDBwEBAwIDQQAbja0Q41/rIKOOvjHKsAsoEJM+WJf6  
 /PKXg2JaStthmw99bdtwkwU/qDbcje2tF6mt+XWyQBXwvfeEs1GFY9fJ

A.5 Данные получателя с ключом длины 512 бит:

- ключ ЭП получателя с ключом длины 512 бит:

A50315981F0A7C7FC05B4EB9591A62B1F84BD6FD518ACFCEDF0A7C9CF388D1F18757  
 C056ADA5B38CBF24CDB0F1519EF72DB1712CEF1920952E94AF1F9C575DC<sub>16</sub>

- идентификатор кривой получателя с ключом длины 512 бит:

1.2.643.7.1.2.1.2.1

- сертификат получателя с ключом длины 512 бит, представленный в кодировке BASE64:

MIICNTCCAEKgAwIBAgIEAYy6hTAKBggqhQMHAQEDAja4MQ0wCwYDVQQKEwRUSzI2  
 MSCwJQYDVQQDEX5DQSBUSzI2OiBHT1NUIDM0LjEwLTEyIDI1Ni1iaXQwHhcNMDEw  
 MTAxMDAwMDAwWhcNNDkxMjMxMDAwMDAwWjA6MQ0wCwYDVQQKEwRUSzI2MSkwJwYD  
 VQQDEyBSRUNJUE1FT1Q6IEDPU1QgMzQuMTAtMTIgNTEyLWJpdDCBqjAhBggqhQMH  
 AQEBAjAVBqkqhQMHAQIBAgEGCCqFAwcBAQIDA4GEAASBgKauwGYvUkz19g0LP/p  
 zeRdmwy1m+QSy9W5ZrL/AGuJofm2ARjz40ozNbW6bp9hkHu8x66LX7u5zz+QeS2+  
 X5om18UXriComg0+qhbzc+Hzu0eQ8FjOd8LpLk3TzZfBltfLOX5iPLjeum+pSP  
 0QjoXAVcrop//B4yvZIukvRoo4GFMIGCMGEGA1UdAQRaMFiAFIDZDPeZ+GZNk1OJ  
 jsCecS2npzESoTowODENMAsGA1UEChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsyNjog  
 R09TVCAzNC4xMC0xMiAyNTYtYml0ggQBjLqBMB0GA1UdDgQWBBSrXT5VKhm/5uff  
 kwW0XpG19k6AajAKBggqhQMHAQEDAgnBAAJBpsHRrQKZGb22LozaReEB8rl2MbIR  
 ja64NaM5h+cAFoHm6t/k+zilh2A11rTakR+5of4NQ3EjEhuPtomP2tc=

## A.6 Примеры содержимого типа «подписанные данные»

### A.6.1 Содержимое типа «подписанные данные» с атрибутами

В примере ниже использованы данные получателя и отправителя с ключами длины 512 бит.

A.6.1.1 Содержимое типа «подписанные данные» с атрибутами в кодировке BASE64

MIIEwYJKoZIhvCNQcCoIEKDCCBCQCAQExDDAKBggqhQMHAQECAzA7BqkqhkiG  
 9w0BBwGgLgQsyu7t8vDu6/zt++kg7/Do7OXwIOTr/yDx8vDz6vLz8PsgU2lnbmV  
 RGF0YS6gggI6MIICNjCCAeOgAwIBAgIEAYy6hDAKBggqhQMHAQEDAja4MQ0wCwYD  
 VQQKEwRUSzI2MSCwJQYDVQQDEX5DQSBUSzI2OiBHT1NUIDM0LjEwLTEyIDI1Ni1i  
 aXQwHhcNMDEwMTAxMDAwMDAwWhcNNDkxMjMxMDAwMDAwWjA7MQ0wCwYDVQQKEwRU  
 SzI2MSowKAYDVQQDEyFPUK1HSU5BVE9SOiBHT1NUIDM0LjEwLTEyIDUXMi1iaXQw  
 gaowiQYIKoUDBwEBAQIwFQYJKoUDBwECAQIBBggqhQMHAQECAwOBhAAEgYC0i7da  
 vCkOGGVcYqFPtS1fUIROzB0fYARIe0tc1TRpare/qzRuVRapqzzO+K21LDpYVfDP  
 s2Sqa13ZN+Ts/JU1v59qCFB2cYpFyB/0kh4+K79yvz7r8+4WE0EmZf8T3ae/J1Jo  
 6xGunech1/G4hMts9HYLnxbwJDMNVGuIHV6gqzOBhTCBgjBhBgNVHQEEWjBYgBSA

2Qz3mfhmTZNTiY7AnnEtp6cxEqE6MDgxDTALBgNVBAoTBFRMjYxJzA1BgNVBAMT  
 HkNBIFRMLMjY6IEDPU1QgMzQuMTAtMTIgMjU2LWJpdIIIEAYy6gTAdBgNVHQ4EFgQU  
 K+19HAscONGxzCcRpXRAmFHv1XowCgYIKoUDBwEBAwIDQQAbjA0Q41/rIKOOvjHK  
 sAs0EJM+WJf6/PKXg2JaStthmw99bdtwkU/qDbcje2tF6mt+XWyQBXwvfeES1GF  
 Y9fJMYIB1DCCAZACAEwQDA4MQ0wCwyDVQQKEwRUSzI2MScwJQYDVQQDEX5DQSB  
 SzI2OibHT1NUIDM0LjEwLTEYIDI1NiiaXQCBAGMu0QwCgYIKoUDBwEBAgOgga0w  
 GAYJKoZIhvcNAQkDMQsGCSqGSIb3DQEHAcBgkqhkiG9w0BCQUxDxcNMTkwMzIw  
 MTK1NTIyWjAiBgkqhkiG9w0BCWIxFQQTU2lnbmVkIGF0dHIncyB2Ywx1ZTBPB  
 gkqhkig9w0BCQQxQgRAuPHEukF5B1fo9DoQIMdnB0ZLkzq0RueEUZSNv07A7C+GK  
 WiG62fueArg8uPCHPTUN6d/42p33fgMkEwH7f7cDAKBggqhQMHAQEBAgSBgGU  
 nVka8FvTlClmOtj/FUUacBdE/nEBemLOO/535VDYrX1ftPE6zQf/4ghS7TQG2VR  
 GQ3GWD+L3+w09A7d5uyyTEbvgt11UG0OyqFwKmJEaYsMin87SFVs0cn1PGV1fOK  
 eLluza bLx5whxd+mz1pekL5i6ImRX+TpERxrA/xSe5

A.6.1.2 ACH.1 представление содержимого типа «подписанные данные» с атрибутами

```

0 1079: SEQUENCE:
4   9:   OBJECT IDENTIFIER: signedData [1.2.840.113549.1.7.2]
15 1064:   CONTEXT SPECIFIC (0):
19 1060:     SEQUENCE:
23   1:       INTEGER: 1
26   12:       SET:
28   10:         SEQUENCE:
30   8:           OBJECT IDENTIFIER: [1.2.643.7.1.1.2.3]
40  59:         SEQUENCE:
42   9:           OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
53   46:           CONTEXT SPECIFIC (0):
55   44:             OCTET STRING:
:                 CAEEEDF2F0EEEBCFCEDFBE920EFF0E8EC
:                 E5F020E4EBFF20F1F2F0F3EAF2F3F0FB
:                 205369676E6564446174612E
101  570:         CONTEXT SPECIFIC (0):
105  566:           SEQUENCE:
109  483:             SEQUENCE:
113   3:               CONTEXT SPECIFIC (0): INTEGER: 2
118   4:               INTEGER: 26000004
124  10:             SEQUENCE:
126   8:               OBJECT IDENTIFIER: [1.2.643.7.1.1.3.2]
136  56:             SEQUENCE:
138  13:               SET:
140  11:                 SEQUENCE:
142   3:                   OBJECT IDENTIFIER: organizationName [2.5.4.10]
147   4:                   PRINTABLE STRING: 'TK26'
153  39:                 SET:
155  37:                   SEQUENCE:
157   3:                     OBJECT IDENTIFIER: commonName [2.5.4.3]
162  30:                     PRINTABLE STRING:
:                       'CA TK26: GOST 34.10-12 256-bit'
194  30:                   SEQUENCE:
196  13:                     UTC TIME: '010101000000Z'
211  13:                     UTC TIME: '491231000000Z'
226  59:                   SEQUENCE:
228  13:                     SET:
230  11:                       SEQUENCE:
232   3:                         OBJECT IDENTIFIER: organizationName [2.5.4.10]
237   4:                         PRINTABLE STRING: 'TK26'
243  42:                     SET:
245  40:                       SEQUENCE:
247   3:                         OBJECT IDENTIFIER: commonName [2.5.4.3]
252   33:                         PRINTABLE STRING:
```



```

686   64:      SEQUENCE:
688   56:          SEQUENCE:
690   13:              SET:
692   11:                  SEQUENCE:
694    3:                      OBJECT IDENTIFIER: organizationName [2.5.4.10]
699    4:                          PRINTABLE STRING: 'TK26'
705   39:              SET:
707   37:                  SEQUENCE:
709    3:                      OBJECT IDENTIFIER: commonName [2.5.4.3]
714   30:                          PRINTABLE STRING:
718    :                              'CA TK26: GOST 34.10-12 256-bit'
746    4:                  INTEGER: 26000004
752   10:          SEQUENCE:
754    8:              OBJECT IDENTIFIER: [1.2.643.7.1.1.2.3]
764  173:          CONTEXT SPECIFIC (0):
767   24:              SEQUENCE:
769    9:                  OBJECT IDENTIFIER: contentType [1.2.840.113549.1.9.3]
780   11:              SET:
782    9:                  OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
793   28:          SEQUENCE:
795    9:              OBJECT IDENTIFIER: signingTime [1.2.840.113549.1.9.5]
806   15:          SET:
808   13:              UTC TIME: '190320195522Z'
823   34:          SEQUENCE:
825    9:              OBJECT IDENTIFIER: [1.2.840.113549.1.9.98]
836   21:          SET:
838   19:              OCTET STRING:
840    :                  5369676E6564206174747227732076616
842    :                  C7565
859   79:          SEQUENCE:
861    9:              OBJECT IDENTIFIER: messageDigest [1.2.840.113549.1.9.4]
872   66:          SET:
874   64:              OCTET STRING:
876    :                  51D3C712E905E4121FA3D0E840831D9C
878    :                  1D192E4CEAD11B9E11465236FD3B03B0
880    :                  BE18A5A21BAD9FB9E02B83CB8F0873D3
882    :                  50DE9DFF8DA9DF77E03241301FB7FB70
894   10:          SEQUENCE:
896    8:              OBJECT IDENTIFIER: [1.2.643.7.1.1.1.2]
952  128:          OCTET STRING:
954    :                  65275646BC16F4E50A598EB63FC55146
956    :                  9C05D13F9C405E30B38EFF9DF954362B
958    :                  5E57ED3C4EB341FFF88214BB4D01B655
960    :                  1190DC6583F8BDFE5B4F40EDDE6ECB24
962    :                  C46EF82D7659541B43B2A85C0A98911A
964    :                  62C3229FCED2155B34727D4F195D5F38
966    :                  A78B96E65A6CBC79C21C5DFA6CE5A5E9
968    :                  0BE62E889915FE4E9111C6B03FC527B9

```

#### A.6.2 Содержимое типа «подписанные данные» без атрибутов

В примере ниже использованы данные получателя и отправителя с ключами длины 256 бит.

#### A.6.2.1 Содержимое типа «подписанные данные» без атрибутов в кодировке BASE64

```

MIIDAQYJKoZIhvCNQcCoIIC8jCCAu4CAQExDDBAKBggqhQMHAQECAja7BgkqhkiG
9w0BBwGgLgQsyu7t8vDu6/zt++kg7/Do7OXwIOTr/yDx8vDz6vLz8PsgU2lnbmVk
RGF0YS6gggH3MIIB8zCCAaCgAwIBAgIEAYy6gjAKBggqhQMHAQEDAja4MQ0wCwYD
VQQKEwRUSzI2MScwJQYDVQQDEx5DQSBUSzI2OibHT1NUIDM0LjEwLTEyIDI1NiIi
aXQwHhcNMDEwMTAxMDAwMDAwWhcNNdkxMjMxMDAwMDAwWjA7MQ0wCwYDVQQKEwRU
SzI2MSowKAYDVQQDEyFPUk1HSU5BVE9SOiBHT1NUIDM0LjEwLTEyIDI1NiIiaXQw
aDAhBggqhQMHAQEBAxABVgkqhQMHAQIBAQEGCCqFAwcBAQICA0MABECWKQ0TYllq

```

g4GmY3tBJiyzpXUN+aOV9WbmTUinqrmEHP7KCNzoAzFg+04SSQpNNSHpQnm+jLAZ  
 huJaJfqZ6VbTo4GFMIGCMGEA1UdAQRaMFIAFIDZDPeZ+GZNk1OJjsCecS2npzES  
 oTowODENMASGA1UEChMEVEsyNjEnMCUGA1UEAxMeQ0EgVEsyNjogR09TVCAzNC4x  
 MC0xMiAyNTYtYml0ggQBjLqBMB0GA1UdDgQWBTRnChHSWbQYwnJC62n2zu5Njd0  
 3zAKBggqhQMHAQEDAgNBAB41oijaXSeN58178y2rhxY35/1KEq4XWZ70FtsN1VxW  
 ATyzgO5Wliwnt1O4GoZsxx8r6T/i7VG65UNmQ1wdOKQxgaIwgZ8CAQEwQDA4MQ0w  
 CwYDVQQKEwRUSzI2MSCwJQYDVQQDEx5DQSBUSzI2OiBHT1NUIDM0LjEwLTEyIDI1  
 NiliaXQCBAGMuIwCgYIKoUDBwEBAgIwCgYIKoUDBwEBAQEEQC6jZPA59szL9FiA  
 0wC71EBE42ap6gKxkLT800cu2FvbLu972GJYNSI7+UeanVU37OVWyenEXi2E5hku  
 94kBBe8Q=

#### A.6.2.2 ACH.1 представление содержимого типа «подписанные данные» без атрибутов

```

  0  769: SEQUENCE:
  4   9:   OBJECT IDENTIFIER: signedData [1.2.840.113549.1.7.2]
 15  754:   CONTEXT SPECIFIC (0):
 19  750:     SEQUENCE:
 23   1:       INTEGER: 1
 26  12:       SET:
 28  10:         SEQUENCE:
 30   8:           OBJECT IDENTIFIER: [1.2.643.7.1.1.2.2]
 40  59:         SEQUENCE:
 42   9:           OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
 53  46:         CONTEXT SPECIFIC (0):
 55  44:           OCTET STRING:
    :             CAEEEDF2F0EEEBCEDFB920EFF0E8EC
    :             E5F020E4EBFF20F1F2F0F3EAF2F3F0FB
    :             205369676E6564446174612E
101  503:         CONTEXT SPECIFIC (0):
105  499:           SEQUENCE:
109  416:             SEQUENCE:
113   3:               CONTEXT SPECIFIC (0):
115   1:                 INTEGER: 2
118   4:                 INTEGER: 26000002
124  10:                 SEQUENCE:
126   8:                   OBJECT IDENTIFIER: [1.2.643.7.1.1.3.2]
136  56:                 SEQUENCE:
138  13:                   SET:
140  11:                     SEQUENCE:
142   3:                       OBJECT IDENTIFIER: organizationName [2.5.4.10]
147   4:                         PRINTABLE STRING: 'TK26'
153  39:                     SET:
155  37:                       SEQUENCE:
157   3:                         OBJECT IDENTIFIER: commonName [2.5.4.3]
162  30:                         PRINTABLE STRING:
    :                           'CA TK26: GOST 34.10-12 256-bit'
194  30:                     SEQUENCE:
196  13:                       UTC TIME: '010101000000Z'
211  13:                       UTC TIME: '491231000000Z'
226  59:                     SEQUENCE:
228  13:                       SET:
230  11:                         SEQUENCE:
232   3:                           OBJECT IDENTIFIER: organizationName [2.5.4.10]
237   4:                             PRINTABLE STRING: 'TK26'
243  42:                           SET:
245  40:                             SEQUENCE:
247   3:                               OBJECT IDENTIFIER: commonName [2.5.4.3]
252  33:                               PRINTABLE STRING:
    :                                 'ORIGINATOR: GOST 34.10-12 256-bit'
287 104:                     SEQUENCE:

```

```

289   33:           SEQUENCE:
291     8:             OBJECT IDENTIFIER: [1.2.643.7.1.1.1]
301   21:             SEQUENCE:
303     9:               OBJECT IDENTIFIER: [1.2.643.7.1.2.1.1]
314     8:               OBJECT IDENTIFIER: [1.2.643.7.1.1.2.2]
324   67:             BIT STRING UnusedBits:0:
327   64:               OCTET STRING:
330     :                 96290D1362596A8381A6637B41262CB3
331     :                 A5750DF9A395F566E64D48A7AAB9841C
332     :                 FECA08DCE8033160FB4E12490A4D3521
333     :                 E94279BE8CB01986E25A25FA99E956D3
393  133:             CONTEXT SPECIFIC (3):
396  130:               SEQUENCE:
399   97:                 SEQUENCE:
401     3:                   OBJECT IDENTIFIER: authorityKeyIdentifier
[2.5.29.1]
406   90:             OCTET STRING:
408   88:               SEQUENCE:
410   20:                 CONTEXT SPECIFIC (0):
413     :                   80D90CF799F8664D9353898EC09E712D
414     :                   A7A73112
432   58:             CONTEXT SPECIFIC (1):
434   56:               SEQUENCE:
436   13:                 SET:
438   11:                   SEQUENCE:
440     3:                     OBJECT IDENTIFIER:
441     :                         organizationName [2.5.4.10]
442     4:                         PRINTABLE STRING: 'TK26'
451   39:               SET:
453   37:                 SEQUENCE:
455     3:                   OBJECT IDENTIFIER: commonName
[2.5.4.3]
460   30:             PRINTABLE STRING:
461     :                   'CA TK26: GOST 34.10-12 256-bit'
492   4:               CONTEXT SPECIFIC (2):
493     :                   018CBA81
498   29:             SEQUENCE:
500     3:               OBJECT IDENTIFIER: subjectKeyIdentifier
[2.5.29.14]
505   22:             OCTET STRING:
507   20:               OCTET STRING:
508     :                   D19C28474966D06309C90BADA7DB3BB9
509     :                   363774DF
529   10:             SEQUENCE:
531     8:               OBJECT IDENTIFIER: [1.2.643.7.1.1.3.2]
541   65:             BIT STRING UnusedBits:0:
543     :                   1E35A228DA5D2127E7C97BF32DAB8716
544     :                   37E7F94A12AE17599EF416DB0D955C56
545     :                   013CB380EE56962C27B753B81A866CC7
546     :                   1F2BE93FE2ED51BAE54366425C1D38A4
608  162:             SET:
611  159:               SEQUENCE:
614    1:                 INTEGER: 1
617   64:               SEQUENCE:
619   56:                 SEQUENCE:
621   13:                   SET:
623   11:                     SEQUENCE:
625     3:                         OBJECT IDENTIFIER: organizationName [2.5.4.10]

```

```

630      4:          PRINTABLE STRING: 'TK26'
636      39:         SET:
638      37:           SEQUENCE:
640      3:             OBJECT IDENTIFIER: commonName [2.5.4.3]
645      30:             PRINTABLE STRING:
646      :               'CA TK26: GOST 34.10-12 256-bit'
677      4:             INTEGER: 26000002
683      10:            SEQUENCE:
685      8:              OBJECT IDENTIFIER: [1.2.643.7.1.1.2.2]
695      10:            SEQUENCE:
697      8:              OBJECT IDENTIFIER: [1.2.643.7.1.1.1.1]
707      64:            OCTET STRING:
708      :              2EA364F039F6CCCBF45880D300BBD440
709      :              44E366A9EA02B19254FCD3472ED85BDB
710      :              2EEF7BD8625835223BF9479A9D5537EC
711      :              E556C9E9C45E2D84E47914F789017BC4

```

#### A.7 Примеры содержимого типа «конверт данных»

##### A.7.1 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyAgreeRecipientInfo

В примере ниже использованы данные получателя и отправителя с ключами длины 512 бит:

- идентификатор алгоритма шифрования содержимого 1.2.643.7.1.1.5.2.2;
- идентификатор алгоритма шифрования ключа 1.2.643.7.1.1.7.2.1.

##### A.7.1.1 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyAgreeRecipientInfo в кодировке BASE64

```

MIIB/gYJKoZIhvNAQcDoIIB7zCCAesCAQIxggFiOYIBXgIBA6CBo6GBoDAXBggq
hQMHAQEBAjALBgkqhQMHAQIBAgEDgYQABIGAE+itJVNbHM35RHfzuwFJPYdPXqtW
8hNEF7Z/XFEE2T71SRkhFX7ozYKQNh/TkVY9D4vG0LnD9Znr/pJyOjpsNb+dPcKX
Kb/k/0JQxoPGHxFzASVAFq0ov/yBe2XGFWMeKUqtaAr7SvoYS0oEhT5EuT8BXmecd
nRe7NqOzEpb15ahIgQgsqHxOcdOp0311S7k3OH1k1HNa5F8m9ctrOzH2846FMw
FwYJKoUDBwEBBwIBMAoGCCqFAwcBAQYCMHYwdDBAMDgxDTALBgNVBAoTBFRIMjYx
JzAlBgNVBAMTHkNBIFRIMjY6IEDPU1QgMzQuMTAtMTIgMjU2LWJpdAIEAYy6hQQw
SxLc18zMwzLwXbcKqYhV/VzsdbgVARoHsSBIBaThJWE7zI37VGPMQJM5VXJ7GVcL
MF0GCSqGSIB3DQEhATAfBgkqhQMHAQEFAgIwEgQQ6EeVlADDCz2cdEWKy+tM94Av
yIFl/Ie4VeFFuczTsMsIaOUEe3Jn9GeVp8hZSj3O2q4hs1Q/u/+Gj4QkSHm/M0ih
ITAfBgkqhQMHAQAGAQExEgQQs1t6D3J3WCEvxunnEE15NQ==

```

##### A.7.1.2 ACH.1 представление содержимого типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyAgreeRecipientInfo

```

0  510: SEQUENCE:
4   9:   OBJECT IDENTIFIER: envelopedData [1.2.840.113549.1.7.3]
15  495:   CONTEXT SPECIFIC (0):
19  491:     SEQUENCE:
23   1:       INTEGER: 2
26  354:       SET:
30  350:         CONTEXT SPECIFIC (1):
34   1:           INTEGER: 3
37  163:           CONTEXT SPECIFIC (0):
40  160:             CONTEXT SPECIFIC (1):
43   23:               SEQUENCE:
45    8:                 OBJECT IDENTIFIER: [1.2.643.7.1.1.1.2]
55   11:                 SEQUENCE:
57    9:                   OBJECT IDENTIFIER: [1.2.643.7.1.2.1.2.1]
68  132:             BIT STRING UnusedBits:0:
72  128:               OCTET STRING:
73   :                 7BE8AD25535B1CCDF94477F3BB01493D
74   :                 874F5EAB56F2134417B67F5C5104D93E
75   :                 F5491921157EE8CD8290361FD391563D
76   :                 0F8BC6D0B9C3F599EBFE92723A3A6C35

```

```

:
:
:
:
:
203 34: CONTEXT SPECIFIC (1):
205 32: OCTET STRING:
:
:
239 23: SEQUENCE:
241 9:   OBJECT IDENTIFIER: [1.2.643.7.1.1.7.2.1]
252 10:   SEQUENCE:
254 8:     OBJECT IDENTIFIER: [1.2.643.7.1.1.6.2]
264 118: SEQUENCE:
266 116:   SEQUENCE:
268 64:     SEQUENCE:
270 56:       SEQUENCE:
272 13:         SET:
274 11:           SEQUENCE:
276 3:             OBJECT IDENTIFIER: organizationName [2.5.4.10]
281 4:               PRINTABLE STRING: 'TK26'
287 39:             SET:
289 37:               SEQUENCE:
291 3:                 OBJECT IDENTIFIER: commonName [2.5.4.3]
296 30:               PRINTABLE STRING:
:
328 4:                 'CA TK26: GOST 34.10-12 256-bit'
334 48:               OCTET STRING:
:
:
:
384 93:   SEQUENCE:
386 9:     OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
397 31:     SEQUENCE:
399 9:       OBJECT IDENTIFIER: [1.2.643.7.1.1.5.2.2]
410 18:     SEQUENCE:
412 16:       OCTET STRING:
:
430 47:         CONTEXT SPECIFIC (0):
:
:
:
479 33:   CONTEXT SPECIFIC (1):
481 31:     SEQUENCE:
483 9:       OBJECT IDENTIFIER: [1.2.643.7.1.0.6.1.1]
494 18:     SET:
496 16:       OCTET STRING:
:

```

**A.7.2 Содержимое типа «конверт данных» с использованием статического алгоритма согласования ключей KeyAgreeRecipientInfo**

В примере ниже использованы данные получателя и отправителя с ключами длины 256 бит:

- идентификатор алгоритма шифрования содержимого 1.2.643.7.1.1.5.1.1;
- идентификатор алгоритма шифрования ключа 1.2.643.7.1.1.7.1.1.

**A.7.2.1 Содержимое типа «конверт данных» с использованием статического алгоритма согласования ключей KeyAgreeRecipientInfo в кодировке BASE64**

```
MIIBawYJKoZIhvNAQcDoIIBXDCCAVgCAQIxqfehgfQCAQOgQjBAMDgxDTALBgNV
BAoTBFRMjYxJzAlBgNVBAMTHkNBIFRLMjY6IEDPU1QgMzQuMTAtMTIgMjU2LWJp
dAIEAYy6ggEiBCBvcfyuSF57y8vVyaw8Z0ch3wjC41PKTrpVRXty4Rhk5DAXBgkq
hQMHAQEwCgYIKoUDBwEBBgEwbjBsMEAwDENMASGA1UEChMEVEsyNjEnMCUG
```

A1UEAxMeQ0EgVEsyNjogR09TVCAzNC4xMC0xMiAyNTYtYml0AgQBjLqDBChPbi6B  
krXuLPexPAL2oUGCFWDGQHqINL5ExuMBG7/5XQRqriKARVa0MFkGCSqGSIB3DQEHE  
ATAbBgkqhQMHAQEFAQEWdQMDNdCKnYAAA AwqTEDgC9O2bYyTGQJ8WUQGq0zHwzX  
L0jFhWHTF1tcAxYmd9pX5i89UwIxhtYqyjX1QHju2g==

**A.7.2.2 ACH.1 представление содержимого типа «конверт данных» с использованием статического алгоритма согласования ключей KeyAgreeRecipientInfo**

```

0 363: SEQUENCE:
4   9:   OBJECT IDENTIFIER: envelopedData [1.2.840.113549.1.7.3]
15 348:   CONTEXT SPECIFIC (0):
19 344:     SEQUENCE:
23   1:       INTEGER: 2
26 247:     SET:
29 244:       CONTEXT SPECIFIC (1):
32   1:         INTEGER: 3
35 66:         CONTEXT SPECIFIC (0):
37 64:           SEQUENCE:
39 56:             SEQUENCE:
41 13:               SET:
43 11:                 SEQUENCE:
45   3:                   OBJECT IDENTIFIER: organizationName [2.5.4.10]
50   4:                     PRINTABLE STRING: 'TK26'
56 39:               SET:
58 37:                 SEQUENCE:
60   3:                   OBJECT IDENTIFIER: commonName [2.5.4.3]
65 30:                     PRINTABLE STRING:
:                       'CA TK26: GOST 34.10-12 256-bit'
97   4:                     INTEGER: 26000002
103 34:               CONTEXT SPECIFIC (1):
105 32:                 OCTET STRING:
:                   6F71FCAE485E7BCBCBD5C9AC3C674721
:                   DF08C2E253CA4EBA55457B72E11864E4
139 23:               SEQUENCE:
141   9:                 OBJECT IDENTIFIER: [1.2.643.7.1.1.7.1.1]
152 10:                 SEQUENCE:
154   8:                   OBJECT IDENTIFIER: [1.2.643.7.1.1.6.1]
164 110:               SEQUENCE:
166 108:                 SEQUENCE:
168 64:                   SEQUENCE:
170 56:                     SEQUENCE:
172 13:                       SET:
174 11:                         SEQUENCE:
176   3:                           OBJECT IDENTIFIER: organizationName [2.5.4.10]
181   4:                             PRINTABLE STRING: 'TK26'
187 39:                           SET:
189 37:                             SEQUENCE:
191   3:                               OBJECT IDENTIFIER: commonName [2.5.4.3]
196 30:                                 PRINTABLE STRING:
:                                   'CA TK26: GOST 34.10-12 256-bit'
228   4:                                 INTEGER: 26000003
234 40:                               OCTET STRING:
:                                 4F6E2E8192B5EE2CF7B13C02F6A14182
:                                 1560C6407A8834BE44C6E3011BBFF95D
:                                 046AAE22804556B4
276 89:               SEQUENCE:
278   9:                 OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
289 27:                 SEQUENCE:
291   9:                   OBJECT IDENTIFIER: [1.2.643.7.1.1.5.1.1]
302 14:                     SEQUENCE:

```

```
304 12:          OCTET STRING:  
      :          74D7422A7600000030A93103  
318 47:          CONTEXT SPECIFIC (0):  
      :          4ED9B6324C6409F165101AAD331F0CD7  
      :          2F48C58561D3175B5C03162677DA57E6  
      :          2F3D53023186D62ACA35F54078EEDA
```

**A.7.3 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo**

В примере ниже использованы данные получателя и отправителя с ключами длины 256 бит:

- идентификатор алгоритма шифрования содержимого 1.2.643.7.1.1.5.2.1;
- идентификатор алгоритма шифрования ключа 1.2.643.7.1.1.7.2.1.

**A.7.3.1 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo в кодировке BASE64**

```
MIIB1QYJKoZIhvcNAQcDoIIBhjCCAYICAQAxggEcMIIBGAIBADBADgxDTALBgNV  
BAoTBFRLMjYxJzA1BgNVBAMTHkNBIFRJMjY6IEDPU1QgMzQuMTAtMTIgMjU2LWJp  
dAIEAYy6gzAXBgkqhQMHAQEHAgEwCgYIKoUBBwEBBgEEgbcwgbQEMFiMredFR3Mv  
3g2wqyVXRnrbYEBCNFaqggBpHwPQh3bF98t+9HZPxRDCww0OPfxeuTBeMBcGCCqF  
AwcBAQEBMAsGCSqFAwcBAgeBAQNDARAdFJ9ww+3ptvQiaQpizCldNYh14DB1r18  
Fx/2FIgnwssCbYRQ+UuRsTk9dfLLTGJG3JIEXKFxXWBgOrK965A5pAQg9f2/EHxG  
DfetwCe1a6uUDCWD+wp5dYOpfkry8YRDEJgwXQYJKoZIhvcNAQcBMB8GCSqFAwcB  
AQUCATASBBUHNxmVcLO/v30aY9P7jxOgC+sD9CHG1EMRUpfGn6yfFDMExmYeby8  
LzdPJelMkYY0qQgdC1zI3nQ7/4taf+4zRA==
```

**A.7.3.2 ACH.1 представление содержимого типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo**

```
0 405: SEQUENCE:  
4   9:   OBJECT IDENTIFIER: envelopedData [1.2.840.113549.1.7.3]  
15 390:   CONTEXT SPECIFIC (0):  
19 386:     SEQUENCE:  
23  1:       INTEGER: 0  
26 284:       SET:  
30 280:         SEQUENCE:  
34  1:           INTEGER: 0  
37 64:           SEQUENCE:  
39 56:             SEQUENCE:  
41 13:               SET:  
43 11:                 SEQUENCE:  
45  3:                   OBJECT IDENTIFIER: organizationName [2.5.4.10]  
50  4:                     PRINTABLE STRING: 'TK26'  
56 39:               SET:  
58 37:                 SEQUENCE:  
60  3:                   OBJECT IDENTIFIER: commonName [2.5.4.3]  
65 30:                     PRINTABLE STRING:  
   :                       'CA TK26: GOST 34.10-12 256-bit'  
97  4:           INTEGER: 26000003  
103 23:           SEQUENCE:  
105  9:             OBJECT IDENTIFIER: [1.2.643.7.1.1.7.2.1]  
116 10:             SEQUENCE:  
118  8:               OBJECT IDENTIFIER: [1.2.643.7.1.1.6.1]  
128 183:             OCTET STRING:  
131 180:               SEQUENCE:  
134 48:                 OCTET STRING:  
   :                   588CADE74547732FDE0DB0AB2557467A  
   :                   E160404C3456AAAA00691F03D08776C5  
   :                   F7CB6DF4764FC510C2C30D0E3DFC5EB9  
184  94:               SEQUENCE:  
186 23:                 SEQUENCE:  
188  8:                   OBJECT IDENTIFIER: [1.2.643.7.1.1.1.1]
```

```

198   11:      SEQUENCE:
200     9:          OBJECT IDENTIFIER: [1.2.643.7.1.2.1.1.1]
211    67:          BIT STRING UnusedBits:0:
214    64:              OCTET STRING:
215     :                  74527DC30FB7A6DBD089A4298B30A574
216     :                  D6219780C1D6B97C171FF6148827C2CB
217     :                  026D8450F94B91B1393D75F2CB4C6246
218     :                  DC92045CA1715D60603AB2BDEB9039A4
280    32:          OCTET STRING:
281     :                  F5FDBF107C460DF7ADC027B56BAB940C
282     :                  2583FB0A797583A97E4AF2F184431098
314    93:      SEQUENCE:
316     9:          OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
327    31:          SEQUENCE:
329     9:              OBJECT IDENTIFIER: [1.2.643.7.1.1.5.2.1]
340    18:          SEQUENCE:
342    16:              OCTET STRING:
343     :                  D41CDC6655C94EFEFDCE698F4FEE3C4E
360    47:          CONTEXT SPECIFIC (0):
361     :                  AC0FD0871A510C454A5F1A7EB27C50CC
362     :                  13199879BCBC2F374F25ED4C918574A9
363     :                  081D0B5CC8DE743BFF8B5A7FEE3344

```

#### **A.7.4 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo**

В примере ниже использованы данные получателя и отправителя с ключами длины 512 бит:

- идентификатор алгоритма шифрования содержимого 1.2.643.7.1.1.5.1.2;
- идентификатор алгоритма шифрования ключа 1.2.643.7.1.1.7.1.1.

#### **A.7.4.1 Содержимое типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo в кодировке BASE64**

```

MIIB5wYJKoZIhvNAQcDoIIB2DCCAdQCAQAxggFXMIIBuWIBADMgxDTALBgNV
BAoTBFRLMjYxJzAlBgNVBAMThkNBIFRLMjY6IEDPU1QgMzQuMTAtMTIgMjU2LWJp
dAIEAYy6hTAXBgkqhQMHAQEHAQEWGgYIKoUDBwEBBgIEgfIwge8EKLgwbJFP21Qe
yKTSzdBqEqvb59bsbZ+xF5+s2zo0vKNZTYuoIkG7Ks4wgaAwFwYIKoUDBwEBAQIw
CwYJKoUDBwECAQIBA4GEAASBgNPm7e14dxH70hcJhp3PJjBgj45ptP8DB5Cvt2jd
PCnFCOQUugmkMXcDEGoaYiPMicjSJvhZhyOQuwiTveEzRFQE2qLbK51nUJOa0BC9
/4G1v5t79iholr1xB5fc4cduy29YE/tb0CU9o14ZyVJdmqzLaSwjDtqLBh+jqD54
KpVJBCBozTnW69sZQBrqTqxzQI3j4QCpUA1my8IYxUtxlni3pjBZBgkqhkiG9w0B
BwEwGwYJKoUDBwEBBQECMA4EDNmoGe4ZUZlF98u8x4AvoyuvFwsqPRN5DQn+obUB
gfrwMYfqmeZ34MAkb3QYcFyck7+kptHr8wxlo/6/mNGhGTAXBgkqhQMHAQAGAQEx
CgQIyyOpgBfMHxM=

```

#### **A.7.4.2 ACH.1 представление содержимого типа «конверт данных» с использованием эфемерного алгоритма согласования ключей KeyTransRecipientInfo**

```

0  487:  SEQUENCE:
4   9:      OBJECT IDENTIFIER: envelopedData [1.2.840.113549.1.7.3]
15  472:      CONTEXT SPECIFIC (0):
19  468:          SEQUENCE:
23   1:              INTEGER: 0
26  343:          SET:
30  339:              SEQUENCE:
34   1:                  INTEGER: 0
37   64:              SEQUENCE:
39   56:              SEQUENCE:
41   13:                  SET:
43   11:                      SEQUENCE:
45   3:                          OBJECT IDENTIFIER: organizationName [2.5.4.10]
50   4:                              PRINTABLE STRING: 'TK26'
56   39:                  SET:
58   37:                      SEQUENCE:

```

```

60      3:          OBJECT IDENTIFIER: commonName [2.5.4.3]
65      30:          PRINTABLE STRING:
:          :
97      4:          'CA TK26: GOST 34.10-12 256-bit'
103     23:          INTEGER: 26000005
SEQUENCE:
105     9:          OBJECT IDENTIFIER: [1.2.643.7.1.1.7.1.1]
116     10:          SEQUENCE:
118     8:          OBJECT IDENTIFIER: [1.2.643.7.1.1.6.2]
128    242:          OCTET STRING:
131    239:          SEQUENCE:
134    40:          OCTET STRING:
:          B8306C914FDB541EC8A4D2CDD06A12AB
:          DBE7D6EC6D9FB1179FACD99A34BCA359
:          4D8BA82241BB2ACE
176   160:          SEQUENCE:
179   23:          SEQUENCE:
181     8:          OBJECT IDENTIFIER: [1.2.643.7.1.1.1.2]
191    11:          SEQUENCE:
193     9:          OBJECT IDENTIFIER: [1.2.643.7.1.2.1.2.1]
204   132:          BIT STRING:
208   128:          OCTET STRING:
:          D3E6EDED787571FBD21089869DCF2630
:          608F8E69B4FF030790AFB768C33C29C5
:          08E414BA09A4317703106A1A6223CC89
:          C8D226F859872390BB0893BDE1334454
:          04DAA2DB2B9D6750939AD010BDFF81B5
:          BF9B7BF628682EBD710797DCE1C76ECB
:          6F5813FB5BD0253DA35E19C9525D9AAC
:          CB692C230EDA8B061FA3A83E782A9549
339   32:          OCTET STRING:
:          4ECD39D6EBDB19401AEA4EAC73408DE3
:          E100A9500D66CBC218C54B719678B7A6
373   89:          SEQUENCE:
375     9:          OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
386   27:          SEQUENCE:
388     9:          OBJECT IDENTIFIER: [1.2.643.7.1.1.5.1.2]
399   14:          SEQUENCE:
401   12:          OCTET STRING:
:          D9A819EE19519945F7CBBC7
415   47:          CONTEXT SPECIFIC (0):
:          A32BAF170B2A3D13790D09FEA1B50181
:          FAF03187EA99E677E0C0246F7418705C
:          9C93BFA4A6D1EBF30C653BFEBF98D1
464   25:          CONTEXT SPECIFIC (1):
466   23:          SEQUENCE:
468     9:          OBJECT IDENTIFIER: [1.2.643.7.1.0.6.1.1]
479   10:          SET:
481     8:          OCTET STRING:
:          CB23A98017CC1F13

```

#### A.8 Примеры содержимого типа «хэшированные данные»

##### A.8.1 Содержимое типа «хэшированные данные» с длиной хэш-кода 256 бит

В примере ниже представлено содержимое типа «хэшированные данные» с длиной хэш-кода 256 бит.

###### A.8.1.1 Содержимое типа «хэшированные данные» в кодировке BASE64

```
MH0GCSqGSIb3DQEHBaBwMG4CAQAwCgYIKoUDBwEBAgIwOwYJKoZIhvCNQcBoC4E
LMru7fLw7uv87fvpIO/w6Oz18CDk6/8g8fLw8+ry8/D7IERpZ2VzdERhdGEuBCD/
esPQYsGkzxZV8uUMIAWt6SI8KtxBP8NyG8AGbJ8i/Q==
```

**A.8.1.2 ACH.1 представление содержимого типа «хэшированные данные»**

```

0 125: SEQUENCE:
2   9:   OBJECT IDENTIFIER: digestData [1.2.840.113549.1.7.5]
13 112:   CONTEXT SPECIFIC (0):
15 110:     SEQUENCE:
17   1:       INTEGER: 0
20   10:     SEQUENCE:
22   8:       OBJECT IDENTIFIER: [1.2.643.7.1.1.2.2]
32   59:     SEQUENCE:
34   9:       OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
45   46:     CONTEXT SPECIFIC (0):
47   44:       OCTET STRING:
          :
          CAEEEDF2F0EEEBCEDFBE920EFF0E8EC
          :
          E5F020E4EBFF20F1F2F0F3EAF2F3F0FB
          :
          20446967657374446174612E
93   32:     OCTET STRING:
          :
          FF7AC3D062C1A4CF1655F2E50C2005AD
          :
          E9223C2ADC413FC3721BC0066C9F22FD

```

**A.8.2 Содержимое типа «хэшированные данные» с длиной хэш-кода 512 бит**

В примере ниже представлено содержимое типа «хэшированные данные» с длиной хэш-кода 512 бит.

**A.8.2.1 Содержимое типа «хэшированные данные» в кодировке BASE64**

MIGfBgkqhkiG9w0BBwWggZEwgY4CAQAwCgYIKoUDBwEBAgMwOwYJKoZIhvcNAQcB  
 oC4ELMu7fLw7uv87fvPIO/w6Oz18CDk6/8g8fLw8+rly8/D7IERpZ2VzdERhdGEu  
 BEDe4VUvcKSRvU7RFvhFja+jXY+nJSUkUsoi3oOeJBnru4PErt8RusPrCJs614ciH  
 CM+ehrC4a+M1Nbq77F/Wsa/v

**A.8.2.2 ACH.1 представление содержимого типа «хэшированные данные»**

```

0 159: SEQUENCE:
3   9:   OBJECT IDENTIFIER: digestData [1.2.840.113549.1.7.5]
14 145:   CONTEXT SPECIFIC (0):
17 142:     SEQUENCE:
20   1:       INTEGER: 0
23   10:     SEQUENCE:
25   8:       OBJECT IDENTIFIER: [1.2.643.7.1.1.2.3]
35   59:     SEQUENCE:
37   9:       OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
48   46:     CONTEXT SPECIFIC (0):
50   44:       OCTET STRING:
          :
          CAEEEDF2F0EEEBCEDFBE920EFF0E8EC
          :
          E5F020E4EBFF20F1F2F0F3EAF2F3F0FB
          :
          20446967657374446174612E
96   64:     OCTET STRING:
          :
          DEE1552F70A491BD4ED11558458DA8D7
          :
          63E9C9494914B288B7A0E789067AEEE0
          :
          F12BB7C46EB0FAC226CEB5E1C88708CF
          :
          9E86B0B86BE33535BABEC5FD6B1AFEF

```

**A.9 Примеры содержимого типа «зашифрованные данные»**

В следующих примерах для шифрования содержимого использован ключ:

8F5EEF8814D228FB2BBC5612323730CFA33DB7263CC2C0A01A6C6953F33D61D5<sub>16</sub>

**A.9.1 Содержимое типа «зашифрованные данные» для алгоритма «Магма»**

Идентификатор алгоритма шифрования содержимого: 1.2.643.7.1.1.5.1.2.

**A.9.1.1 Содержимое типа «зашифрованные данные» в кодировке BASE64**

MIGIBgkqhkiG9w0BBwagezB5AgEAMFkGCSqGSIB3DQEHAAbBgkqhQMHAQEFAQIw  
 DgQMuncOu3uYPbI30vFCgC9Nsws4R09yLp6jUtadncWUPZGmCGpPKnXGgNHvEmUA  
 rgKJvu4FPhtLkHuLeQXZg6EZMBcGCSqFAwcBAAYBATEKBAjCbQoH632oGA==

A.9.1.2 ACH.1 представление содержимого типа «зашифрованные данные»

```
0 136: SEQUENCE:
  3   9:   OBJECT IDENTIFIER: encryptedData [1.2.840.113549.1.7.6]
  14  123: CONTEXT SPECIFIC (0):
  16  121:   SEQUENCE:
  18   1:     INTEGER: 0
  21   89:   SEQUENCE:
  23   9:     OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
  34   27:   SEQUENCE:
  36   9:     OBJECT IDENTIFIER: [1.2.643.7.1.1.5.1.2]
  47  14:   SEQUENCE:
  49  12:     OCTET STRING:
  :           BA770EBB7B983DB237D2F142
  63  47:   CONTEXT SPECIFIC (0):
  :           4DB30B38474F722E9EA352D69D9DC594
  :           3D91A6086A4F2A75C680D1EF126500AE
  :           0289BEEE053C7B4B907B8B7905D983
  112 25: CONTEXT SPECIFIC (1):
  114 23:   SEQUENCE:
  116   9:     OBJECT IDENTIFIER: [1.2.643.7.1.0.6.1.1]
  127  10:   SET:
  129   8:     OCTET STRING:
  :           C26D0A07EB7DA818
```

**A.9.2 Содержимое типа «зашифрованные данные» для алгоритма «Кузнецик»**

Идентификатор алгоритма шифрования содержимого: 1.2.643.7.1.1.5.2.1.

**A.9.2.1 Содержимое типа «зашифрованные данные» в кодировке BASE64**

MHEGCSqGSIB3DQEHBqBkMGICAQAwXQYJKoZIhvcNAQcBMB8GCSqFAwcBAQUCATAS  
BBBSwX+zyOEPPuGyfpsRG4AigC/P8ftTdQMStfITHvKE/vpJlwaHgGv83m2bsPay  
eyuaprpTeEMOaqGcO0MxHWsc9hQ==

**A.9.2.2 ACH.1 представление содержимого типа «зашифрованные данные»**

```
0 113: SEQUENCE:
  2   9:   OBJECT IDENTIFIER: encryptedData [1.2.840.113549.1.7.6]
  13  100: CONTEXT SPECIFIC (0):
  15  98:   SEQUENCE:
  17   1:     INTEGER: 0
  20  93:   SEQUENCE:
  22   9:     OBJECT IDENTIFIER: data [1.2.840.113549.1.7.1]
  33  31:   SEQUENCE:
  35   9:     OBJECT IDENTIFIER: [1.2.643.7.1.1.5.2.1]
  46  18:   SEQUENCE:
  48  16:     OCTET STRING:
  :           52C17FB3C8E10F3EE1B27E9B111B8022
  66  47:   CONTEXT SPECIFIC (0):
  :           CFF1FB53750312B5F213855904FEFA49
  :           970687806BFCDE6D9BB0F6B27B2BAAA6
  :           84DE10C39AA8670ED0CC475AC0BD85
```

## Библиография

- [1] 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 Р 34.10, ГОСТ Р 34.11 в профиле сертификата и списке отзыва сертификатов (CRL) инфраструктуры открытых ключей X.509. Версия 2
- [3] 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



## **БЗ 10—2019/65**

Редактор *Л.С. Зимилова*  
Технический редактор *В.Н. Прусакова*  
Корректор *М.С. Кабашова*  
Компьютерная верстка *Л.А. Круговой*

Сдано в набор 04.09.2019. Подписано в печать 17.09.2019. Формат 60×84<sup>1</sup>/<sub>8</sub>. Гарнитура Ариал.  
Усл. печ. л. 6,05. Уч.-изд. л. 5,47.  
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

---

Создано в единичном исполнении во ФГУП «СТАНДАРТИНФОРМ» для комплектования Федерального информационного фонда стандартов, 117418 Москва, Нахимовский пр-т, д. 31, к. 2.  
[www.gostinfo.ru](http://www.gostinfo.ru) [info@gostinfo.ru](mailto:info@gostinfo.ru)