Международный союз электросвязи

МСЭ-Т X.1080.0

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

(03/2017)

СЕРИЯ X: СЕТИ ПЕРЕДАЧИ ДАННЫХ,
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
И БЕЗОПАСНОСТЬ

Information and network security – Телебиометрия

Управление доступом для защиты телебиометрических данных

Рекомендация МСЭ-Т X.1080.0

Международный союз электросвязи

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ X

СЕТИ ПЕРЕДАЧИ ДАННЫХ, ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ И БЕЗОПАСНОСТЬ

СЕТИ ПЕРЕДАЧИ ДАННЫХ ОБЩЕГО ПОЛЬЗОВАНИЯ	X.1–X.199
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ	X.200–X.299
ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕТЯМИ	X.300–X.399
СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ	X.400–X.499
СПРАВОЧНИК	X.500–X.599
ОРГАНИЗАЦИЯ СЕТИ ВОС И СИСТЕМНЫЕ АСПЕКТЫ	X.600–X.699
УПРАВЛЕНИЕ В ВОС	X.700–X.799
БЕЗОПАСНОСТЬ	X.800–X.849
ПРИЛОЖЕНИЯ ВОС	X.850–X.899
ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА	X.900–X.999
БЕЗОПАСНОСТЬ ИНФОРМАЦИИ И СЕТЕЙ	
Общие аспекты безопасности	X.1000–X.1029
Безопасность сетей	X.1030–X.1049
Управление безопасностью	X.1050–X.1069
Телебиометрия	**X.1080–X.1099**
БЕЗОПАСНЫЕ ПРИЛОЖЕНИЯ И УСЛУГИ	
Безопасность многоадресной передачи	X.1100–X.1109
Безопасность домашних сетей	X.1110–X.1119
Безопасность подвижной связи	X.1120–X.1139
Безопасность веб-среды	X.1140–X.1149
Протоколы безопасности	X.1150–X.1159
Безопасность одноранговых сетей	X.1160–X.1169
Безопасность сетевой идентификации	X.1170–X.1179
Безопасность IPTV	X.1180–X.1199
БЕЗОПАСНОСТЬ КИБЕРПРОСТРАНСТВА	
Кибербезопасность	X.1200–X.1229
Противодействие спаму	X.1230–X.1249
Управление определением идентичности	X.1250–X.1279
БЕЗОПАСНЫЕ ПРИЛОЖЕНИЯ И УСЛУГИ	
Связь в чрезвычайных ситуациях	X.1300–X.1309
Безопасность повсеместных сенсорных сетей	X.1310–X.1339
Рекомендации, связанные с PKI	X.1340–X.1349
Безопасность интернета вещей (IoT)	X.1360–X.1369
Безопасность интеллектуальный транспортных системы (ИТС)	X.1370–X.1379
ОБМЕН ИНФОРМАЦИЕЙ, КАСАЮЩЕЙСЯ КИБЕРБЕЗОПАСНОСТИ	
Обзор кибербезопасности	X.1500–X.1519
Обмен информацией об уязвимости/состоянии	X.1520–X.1539
Обмен информацией о событии/инциденте/эвристических правилах	X.1540–X.1549
Обмен информацией о политике	X.1550–X.1559
Эвристические правила и запрос информации	X.1560–X.1569
Идентификация и обнаружение	X.1570–X.1579
Гарантированный обмен	X.1580–X.1589
БЕЗОПАСНОСТЬ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ	
Обзор безопасности облачных вычислений	X.1600–X.1601
Проектирование безопасности облачных вычислений	X.1602–X.1639
Передовой опыт и руководящие указания в области облачных вычислений	X.1640–X.1659
Обеспечение безопасности облачных вычислений	X.1660–X.1679
Другие вопросы безопасности облачных вычислений	X.1680–X.1699

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ-Т.

Рекомендация МСЭ-Т X.1080.0

Управление доступом для защиты телебиометрических данных

Резюме

В Рекомендации МСЭ-Т X.1080.0 "Управление доступом для защиты телебиометрических данных" определяются способы защиты телебиометрической информации от несанкционированного доступа. В ней применяется ориентированный на услуги подход, при котором предоставляется лишь та информация, которая необходима для конкретной цели, то есть доступ к информации предоставляется не только на основе *права на информацию*, но и на основании *необходимости* в ней.

Основным элементом настоящей Рекомендации является спецификация атрибута, содержащаяся в сертификате атрибутов или сертификате открытого ключа, которая подробно описывает круг привилегий того или иного объекта/субъекта для одного или нескольких типов услуг.

Безопасность обеспечивается путем использования профиля синтаксиса криптографических сообщений (CMS). Этот профиль обеспечивает аутентификацию и целостность данных, а также при необходимости их конфиденциальность (шифрование).

Профиль CMS призван служить средством поддержки безопасности во всевозможных телебиометрических спецификациях. Этот профиль предполагает наличие правильно развернутой инфраструктуры открытых ключей (PKI) и зависит от правильного развертывания такой инфраструктуры.

Кроме того, применимость настоящей Рекомендации обусловлена развертыванием инфраструктуры управления привилегиями (PMI).

Хронологическая справка

Издание	Рекомендация	Утверждение	Исследовательская комиссия	Уникальный идентификатор*
1.0	МСЭ-Т X.1080.0	30.03.2017 г.	17-я	11.1002/1000/13193

Ключевые слова

Управление доступом, протокол Диффи–Хеллмана, PKI, телебиометрия.

* Для получения доступа к Рекомендации наберите в адресном поле вашего веб-браузера URL: http://handle.itu.int/, после которого укажите уникальный идентификатор Рекомендации. Например, http://handle.itu.int/11.1002/1000/11830-en.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи и информационно-коммуникационных технологий (ИКТ). Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним в целях стандартизации электросвязи на всемирной основе.

На Всемирной ассамблее по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяются темы для изучения Исследовательскими комиссиями МСЭ-Т, которые в свою очередь вырабатывают Рекомендации по этим темам.

Утверждение Рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соблюдение положений данной Рекомендации осуществляется на добровольной основе. Однако данная Рекомендация может содержать некоторые обязательные положения (например, для обеспечения функциональной совместимости или возможности применения), и в таком случае соблюдение Рекомендации достигается при выполнении всех указанных положений. Для выражения требований используются слова "следует", "обязан" (shall) или некоторые другие обязывающие выражения, такие как "должен" (must), а также их отрицательные формы. Употребление таких слов не означает, что от какой-либо стороны требуется соблюдение положений данной Рекомендации.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на вероятность того, что практическое применение или выполнение настоящей Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, действительности или применимости заявленных прав интеллектуальной собственности независимо от того, доказываются ли такие права членами МСЭ или другими сторонами, не относящимися к процессу разработки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ не получил извещения об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для выполнения настоящей Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что вышесказанное может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ по адресу http://www.itu.int/ITU-T/ipr/.

СОДЕРЖАНИЕ

		Стр.
1	Сфера применения	1
2	Справочные документы	1
3	Определения	2
	3.1 Термины, определенные в других документах	2
	3.2 Термины, определенные в настоящей Рекомендации	3
4	Сокращения и акронимы	3
5	Условные обозначения	4
6	Основные понятия и модели	5
	6.1 Защита в рамках одного домена защиты данных	5
	6.2 Защита с пересечением границ доменов защиты данных	6
	6.3 Услугоцентричная модель	7
	6.4 Объектно-атрибутная модель	7
	6.5 Принципы базового управления доступом	7
	6.6 Взаимосвязь с другими схемами управления доступом	8
	6.7 Обзор протоколов	9
	6.8 Использование CMS	9
	6.9 Соображения относительно сертификатов открытых ключей	9
7	Передача информации о привилегиях	9
	7.1 Использование сертификатов атрибутов	9
	7.2 Использование сертификатов открытых ключей	10
	7.3 Тип атрибута accessService	10
	7.4 Операции над объектами в целом	12
	7.5 Операции над атрибутами	13
	7.6 Обработка ошибок	13
8	Протокол заявления привилегий	13
	8.1 Общий обзор	13
	8.2 Общие компоненты запросов	14
	8.3 Доступ к услуге	14
	8.4 Операция чтения	15
	8.5 Операция сравнения	16
	8.6 Операция добавления	17
	8.7 Операция удаления	18
	8.8 Операция изменения	18
	8.9 Операция переименования объекта	20
	8.10 Обработка ошибок	21
	8.11 Выбор информации	21
	8.12 Информация об объекте	22
	8.13 Определенные коды ошибок	22

Стр.

9 Протокол присваивания привилегий 23
9.1 Сфера применения протокола 23
9.2 Типы контента 23
Приложение A − Распределение идентификаторов объектов, используемых в Рекомендациях МСЭ-Т серии X.1080 25
A.1 Верхний уровень дерева идентификаторов объектов 25
A.2 Идентификаторы объектов для типов контента CMS 25
A.3 Идентификаторы объектов для типов атрибутов, относящихся к привилегиям 26
Приложение B − Профиль синтаксиса криптографических сообщений 27
B.1 Общие сведения 27
B.2 Использование типа контента signedData 28
B.3 Использование типа контента envelopedData 30
B.4 Использование типа контента ct-authEnvelopedData 34
B.5 Атрибуты 35
B.6 Коды ошибок синтаксиса криптографических сообщений 36
Приложение C − Формальная спецификация протоколов заявления и присваивания привилегий 38
Добавление I − Неформальная спецификация профиля синтаксиса криптографических сообщений 44
Библиография 49

Введение

При сборе телебиометрических данных частных лиц существует риск нарушения конфиденциальности.

Для защиты такой информации может быть несколько причин. Информация может обеспечить доступ к коммерческой или некоммерческой организации либо носить конфиденциальный характер, влекущий за собой необходимость ограничения ее распространения.

Защита телебиометрических данных от несанкционированного раскрытия имеет два основных аспекта:

– защита данных во время передачи (обычно путем шифрования) и защита хранимых данных;

– управление доступом к хранимым данным.

Телебиометрические системы должны иметь высокий уровень безопасности в части конфиденциальности (шифрования), аутентификации, целостности данных, физической защиты систем, использования брандмауэров, антивирусных программ и т. д., однако в дополнение к этому необходима еще высокоразвитая система управления доступом к хранимой информации, в особенности информации о частных лицах. Последний аспект особенно важен для телебиометрических систем.

Недостаток традиционных систем управления доступом состоит в том, что в них учитывается главным образом *право на доступ* к информации (или отказ в доступе к ней), но не уделяется подробного внимания такому аспекту, как *необходимость* в соответствующей информации. Необходимость в информации подразумевает, что одного только права на доступ к данным недостаточно – должно быть также установлено, что данная информация будет использоваться для какой-либо законной цели.

Информацию следует предоставлять только для тех целей, для которых она предназначена. Медицинская информация о пациенте собирается для того, чтобы обеспечить его оптимальное лечение, и не должна использоваться в других целях, за исключением, возможно, строго контролируемых научно-исследовательских проектов, которые могут потребовать использования некоторой части информации о конкретной группе пациентов. В остальных случаях должна быть обеспечена защита информации от "прочесывания".

Существуют два основных типа управления доступом – физическое и логическое управление. Физическое управление доступом позволяет ограничить доступ на территорию организаций, в здания и помещения, а также к физическим средствам информационных технологий (ИТ). Логическое управление доступом обеспечивает ограничение доступа к компьютерным сетям, системным файлам и данным. В настоящей Рекомендации рассматривается только логическое управление доступом.

Управление доступом включает безопасную аутентификацию субъектов доступа поставщиком услуг. Настоящая Рекомендация предполагает использование цифровых подписей и готовой инфраструктуры открытых ключей (PKI).

Ссылки на Рекомендацию МСЭ-Т X.1080.0 могут содержаться в других биометрических спецификациях.

В Приложении A, которое является неотъемлемой частью настоящей Рекомендации, описывается распределение идентификаторов объектов, используемых в Рекомендациях МСЭ-Т серии X.1080.

Приложение B, которое является неотъемлемой частью настоящей Рекомендации, содержит профиль синтаксиса криптографических сообщений (CMS), соглдасно описанию в IETF RFC 5652 для телебиометрических применений, используемый в данной Рекомендации. Ссылки на него могут также содержаться в других биометрических спецификациях.

Приложение C, которое является неотъемлемой частью настоящей Рекомендации, содержит формальную спецификацию протоколов заявления и присваивания привилегий в форме модуля ASN.1.

Добавление I, которое не является неотъемлемой частью настоящей Рекомендации, содержит неформальную спецификацию профиля CMS в форме модуля абстрактной синтаксической нотации версии 1 (ASN.1).

Рекомендация МСЭ-Т X.1080.0

Управление доступом для защиты телебиометрических данных

1 Сфера применения

Цель настоящей Рекомендации заключается в определении способов защиты конфиденциальности в телебиометрической среде путем использования управления доступом на основе конфиденциальности для телебиометрических применений (АСТ). Хотя в рамках данной Рекомендации нельзя рассмотреть все возможные типы информации, в ее задачи входит предоставление общих инструментов для безопасного обращения с информацией всех типов. Сюда, в частности, входит определение протокола присваивания привилегий и протокола доступа к информации с заявлением привилегий. В настоящей рекомендации содержатся руководящие указания и не соедржатся требования к обеспечению соответствия.

Следующие аспекты выходят за рамки настоящей Рекомендации:

– физическая защита информации;

– несанкционированный доступ производственного персонала, обслуживающего системы безопасности и в связи с этим имеет возможности для ее обхода.

2 Справочные документы

Нижеследующие Рекомендации МСЭ-Т и другие справочные документы содержат положения, которые путем ссылки на них в данном тексте составляют положения настоящей Рекомендации. На момент публикации указанные издания были действующими. Все Рекомендации и другие источники могут подвергаться пересмотру, поэтому всем пользователям данной Рекомендации предлагается рассмотреть возможность применения последнего издания Рекомендаций и других справочных документов, перечисленных ниже. Список действующих на текущий момент Рекомендаций МСЭ-Т регулярно публикуется.

Ссылка на документ, приведенный в настоящей Рекомендации, не придает ему как отдельному документу статус Рекомендации.

[ITU-T X.500-series] Recommendation ITU-T X.5xx (2016) | ISO/IEC 9594-x series, *Information technology – Open Systems Interconnection – The Directory*.

[ITU-T X.501] Recommendation ITU-T X.501 (2016) | ISO/IEC 9594-2, *Information technology – Open Systems Interconnection – The Directory: Models*.

[ITU-T X.509] Recommendation ITU-T X.509 (2016) | ISO/IEC 9594-8, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.

[ITU-T X.520] Recommendation ITU-T X.520 (2016) | ISO/IEC 9594-6, *Information technology – Open Systems Interconnection – The Directory: Selected attribute types*.

[ITU-T X.521] Recommendation ITU-T X.521 (2016) | ISO/IEC 9594-7, *Information technology – Open Systems Interconnection – The Directory: Selected object classes*.

[ITU-T X.680] Recommendation ITU-T X.680 (2015) | ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[ITU-T X.681] Recommendation ITU-T X.681 (2015) | ISO/IEC 8824-2, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.

[ITU-T X.682] Recommendation ITU-T X.682 (2015) | ISO/IEC 8824-3, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.

[ITU-T X.683] Recommendation ITU-T X.683 (2015) | ISO/IEC 8824-4, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

[ITU-T X.690] Recommendation ITU-T X.690 (2015) | ISO/IEC 8825-1, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.

[ITU-T X.1080.1] Recommendation ITU-T X.1080.1 (2011), *e-Health and world-wide telemedicines – Generic telecommunication protocol*.

[ITU-T X.1081] Recommendation ITU-T X.1081 (2011), *The telebiometric multimodal model – A framework for the specification of security and safety aspects of telebiometrics*.

[IETF RFC 2631] IETF RFC 2631 (1999), *Diffie-Hellman Key Agreement Method*.

[IETF RFC 3185] IETF RFC 3185 (2001), *Reuse of CMS Content Encryption Keys*.

[IETF RFC 5083] IETF RFC 5083 (2007), *Cryptographic Message Syntax (CMS) – Authenticated-Enveloped-Data Content Type*.

[IETF RFC 5652] IETF RFC 5652 (2009), *Cryptographic Message Syntax (CMS)*.

[IETF RFC 5753] IETF RFC 5753 (2010), *Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax CMS*.

[IETF RFC 5911] IETF RFC 5911 (2010), *New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME*.

[IETF RFC 6268] IETF RFC 6268 (2011), *Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)*.

3 Определения

3.1 Термины, определенные в других документах

В настоящей Рекомендации используются следующие термины, определенные в других документах.

3.1.1 сертификат атрибутов (attribute certificate) [ITU-T X.509]: Структура данных с цифровой подписью органа атрибутов, связывающая определенные значения атрибута с идентификационной информацией о его держателе.

3.1.2 тип атрибута (attribute type) [ITU-T X.501]: Компонент атрибута, который указывает класс информации, передаваемой атрибутом.

3.1.3 значение атрибута (attribute value) [ITU-T X.501]: Конкретный экземпляр класса информации, указанного типом атрибута.

3.1.4 привилегия (privilege) [ITU-T X.509]: Атрибут или свойство, присвоенное органом объекту.

3.1.5 держатель привилегии (privilege holder) [ITU-T X.509]: Объект, которому присвоена некоторая привилегия. Держатель привилегии может заявлять свою привилегию для конкретной цели.

3.1.6 верификатор привилегий (privilege verifier) [ITU-T X.509]: Объект, проверяющий сертификаты согласно политике привилегий.

3.1.7 источник полномочий (source of authority (SOA)) [ITU-T X.509]: Орган атрибутов, которому верификатор привилегий для конкретного ресурса доверяет как последней инстанции присвоение набора привилегий для доступа к этому ресурсу.

3.2 Термины, определенные в настоящей Рекомендации

В настоящей Рекомендации определены следующие термины.

3.2.1 управление доступом (access control): Метод обеспечения безопасности, используемый для регулирования того, какие субъекты могут выполнять определенные действия с информационными ресурсами в вычислительной среде и какие именно действия.

3.2.2 услуга доступа (access service): Услуга по выполнению конкретной транзакции, предоставляемая поставщиком услуг.

3.2.3 субъект доступа (accessor): Держатель привилегии, который пользуется конкретной услугой доступа с использованием имеющейся у него привилегии.

3.2.4 атрибут (attribute): Некоторое количество информации определенного типа, связанное с объектом. Информация, связанная с объектом, состоит из атрибутов.

3.2.5 домен защиты данных (data protection domain): Домен, в котором подлежащая защите информация находится под одним компонентом управлением.

3.2.6 выделенное имя (distinguished name): Имя, идентифицирующее объект, которое является уникальным в определенном контексте и которое состоит из одного или нескольких компонентов, отражающих положение объекта в иерархии объектов.

3.2.7 объект (object): Какое-либо лицо, подразделение, специалист или некто другой или нечто другое, о котором имеется информация и которое можно идентифицировать по выделенному имени.

3.2.8 класс объектов (object class): Идентифицированное семейство объектов, обладающих определенными общими характеристиками.

3.2.9 операция (operation): Взаимодействие в целях выполнения конкретной задачи, состоящее из запроса со стороны субъекта доступа и ответа со стороны поставщика услуг.

3.2.10 спецификация (specification): Рекомендация МСЭ-Т, международный стандарт или любой устанавливающий документ, разработанный какой-либо признанной организацией по разработке стандартов (ОРС).

4 Сокращения и акронимы

В настоящей Рекомендации используются следующие сокращения и акронимы.

AA	Attribute Authority	Орган атрибутов
ABAC	Attribute Based Access Control	Управление доступом на основе атрибутов
ACL	Access Control List	Список управления доступом
ACT	Access Control for Telebiometrics	Управление доступом для телебиометрических применений
AES	Advanced Encryption Standard	Усовершенствованный стандарт шифрования
ASN.1	Abstract Syntax Notation One	Абстрактная синтаксическая нотация версии 1
BER	Basic Encoding Rules	Базовые правила кодирования
CA	Certification Authority	Орган по сертификации
CEK	Content Encryption Key	Ключ шифрования контента

CMS	Cryptographic Message Syntax		Синтаксис криптографических сообщений
DER	Distinguished Encoding Rules		Отличительные правила кодирования
DH	Diffie-Hellman		Диффи–Хеллман
ECC	Elliptic Curve Cryptography		Шифрование методом эллиптических кривых
ECDH	Elliptic Curve Diffie-Hellman		Протокол Диффи–Хеллмана на эллиптических кривых
GCM	Galois/Counter Mode		Режим счетчика с аутентификацией Галуа
KEK	Key-Encryption Key		Ключ шифрования ключей
LDAP	Lightweight Directory Access Protocol		Облегченный протокол доступа к справочнику (сетевому каталогу)
MAC	Message Authentication Code		Код аутентификации сообщений
PDU	Protocol Data Unit		Блок данных протокола
PKI	Public-Key Infrastructure		Инфраструктура открытых ключей
PMI	Privilege Management Infrastructure		Инфраструктура управления привилегиями
SDO	Standards Developing Organization	OPC	Организация по разработке стандартов
SOA	Source of Authority		Источник полномочий

5 Условные обозначения

В настоящей Рекомендации абстрактная синтаксическая нотация версии 1 (ASN.1) дается полужирным шрифтом **`Courier New`**. Когда типы и значения ASN.1 приводятся в обычном тексте, они выделяются полужирным шрифтом **`Courier New`**.

Если элементы в списке обозначены числами (в противоположность дефисам или буквам), то эти элементы должны рассматриваться как этапы процедуры.

6 Основные понятия и модели

6.1 Защита в рамках одного домена защиты данных

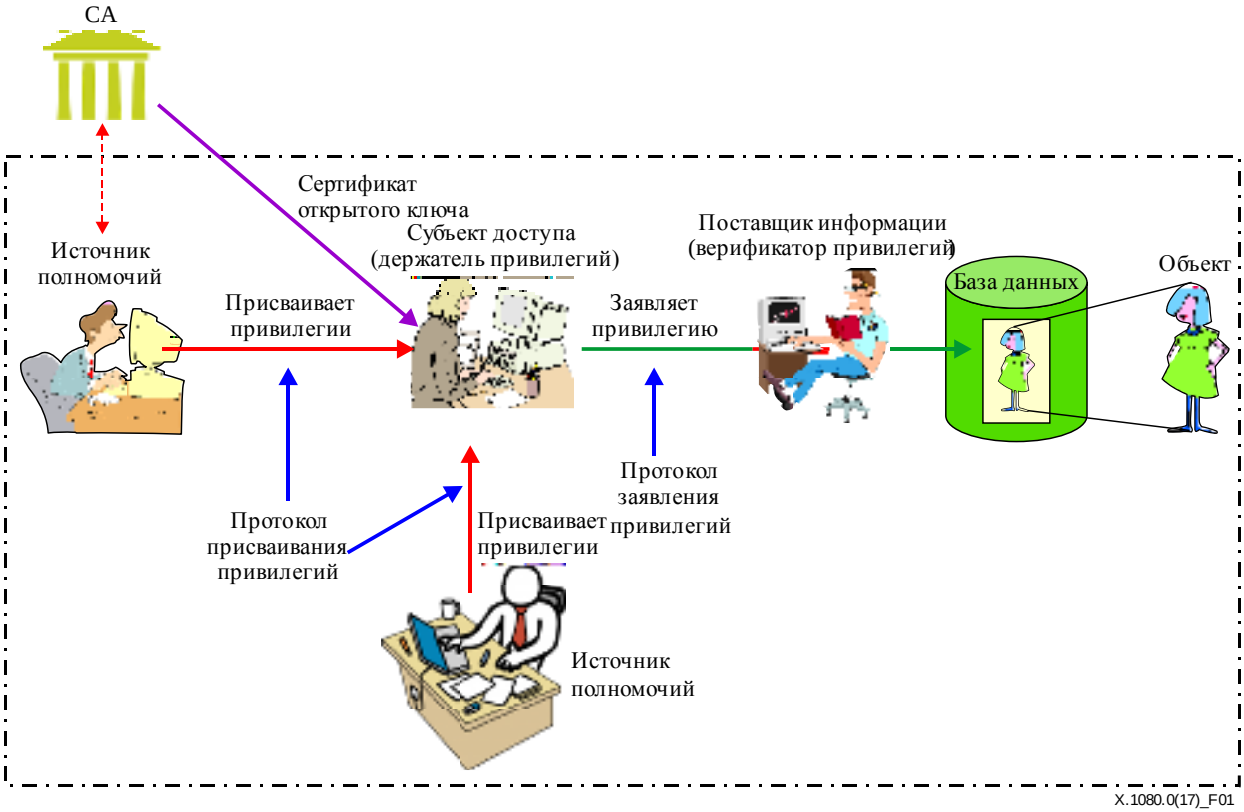

Рисунок 1 − Модель с одним доменом защиты данных

На рисунке 1 показаны различные действующие лица и протоколы в среде управления доступом с одним доменом защиты данных. Домен защиты данных включает в себя все участвующие в защите объекты/субъекты, которые находятся под единым управлением.

У участника под названием "субъект доступа" могут быть должностные обязанности, которые требуют разрешения на выполнение операций с информацией об объектах, хранящейся у поставщика информации. Например, лечащему врачу требуется доступ к медицинской карте пациента, чтобы обеспечить оптимальное лечение, а другим врачам, не обслуживающим данного пациента, эта информация не нужна.

Разрешение на выполнение какой-либо конкретной операции с определенной информацией должно предоставляться только в том случае, если у субъекта доступа есть не только *право* выполнить эту операцию, но и *действительная необходимость* в ней, то есть если ему присвоена соответствующая *привилегия*.

Определение условий, при которых привилегии присваиваются субъектам, выходит за рамки настоящей Рекомендации. В ее задачи входит лишь предоставление необходимых инструментов для безопасного управления привилегиями.

В домене защиты данных должны быть созданы один или несколько органов, уполномоченных присваивать привилегии субъектам. Под термином "источник полномочий" (SOA), введенным в [ITU-T X.509], здесь понимается конечная инстанция по присваиванию привилегий на доступ к конкретной области домена защиты данных.

В настоящей Рекомендации для присваивания привилегий субъектам доступа используются сертификаты. Привилегии могут указываться в компоненте **`attributes`** сертификатов атрибутов или в расширении **`subjectDirectoryAttributes`** сертификатов открытых ключей. Более долгосрочные привилегии указываются, как правило, в сертификатах открытых ключей, а более краткосрочные привилегии – в сертификатах атрибутов.

На рисунке 1 показаны взаимосвязи между различными компонентами одного домена защиты данных. На нем представлены два SOA, каждый из которых отвечает за присваивание привилегий для различных аспектов доступа держателям привилегий.

Одни привилегии могут быть необходимы для повседневных операций субъекта доступа и, следовательно, имеют более долгосрочный характер, а другие присваиваются для особых случаев и поэтому более краткосрочны.

Когда привилегии присваиваются субъекту доступа, он становится держателем привилегий и может пользоваться ими для доступа к информации по протоколу заявления привилегий. Верификатор привилегий, представляющий поставщика информации, проверяет заявляемую привилегию, прежде чем разрешить доступ к информации.

SOA может передавать информацию о привилегиях либо локально, либо путем внедрения этой информации в подписанный сертификат атрибутов в рамках протокола присваивания привилегий, как показано на рисунке 1.

6.2 Защита с пересечением границ доменов защиты данных

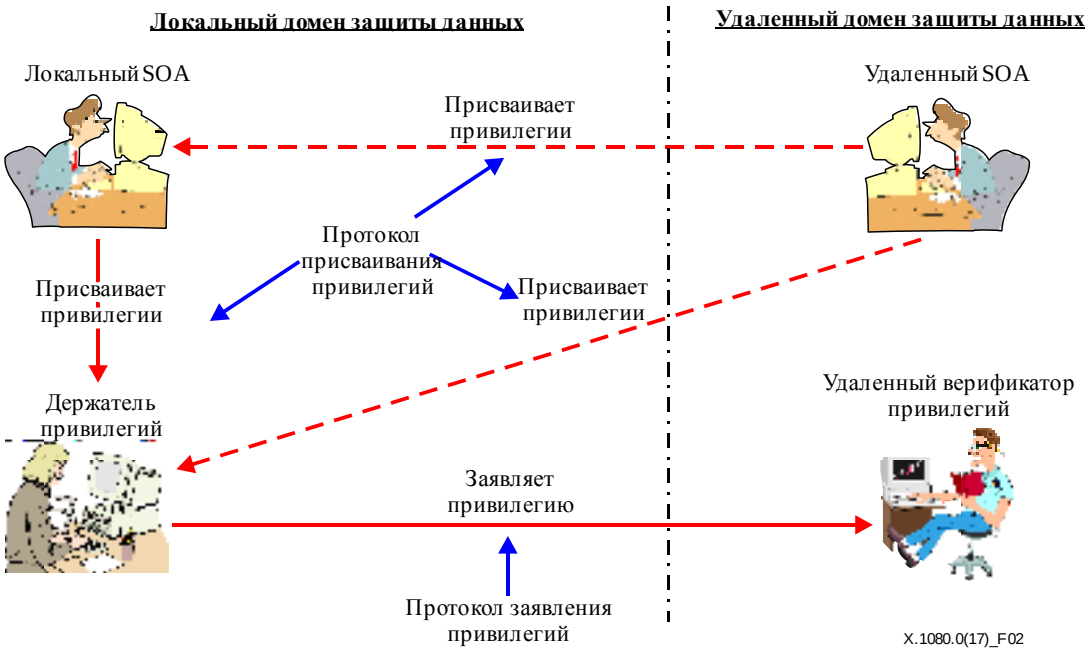

Рисунок 2 – Модель с пересечением границ доменов защиты данных

На рисунке 2 показан случай, когда субъекту доступа требуется доступ к информации в другом домене защиты данных – например для лечения пациента, важные данные анамнеза которого хранятся в другом домене.

В такой среде субъект доступа либо напрямую, либо посредством локального SOA запрашивает доступ к определенной информации о конкретном объекте. Способы передачи этого запроса выходят за рамки настоящей Рекомендации, но могут включать передачу электронной почтой и по телефону.

Удаленный SOA генерирует сертификат атрибутов с атрибутом типа **`accessService`**, содержащим информацию о требуемой привилегии, и подписывает этот сертификат. Компонент **`holder`** сертификата атрибутов может указывать либо на локальный SOA, либо на субъект, которому требуется привилегия.

Если привилегию присваивает локальный SOA, то затем он:

a) при необходимости расшифровывает контент;

b) проверяет подпись контента;

c) извлекает из контента сертификат атрибутов;

d) проверяет извлеченный сертификат атрибутов;

e) создает новый сертификат атрибутов, внеся в него информацию о запрашиваемой привилегии из полученного сертификата атрибутов и указав в качестве держателя объект, которому требуется привилегия;

f) вновь созданный сертификат атрибутов вместе с аналогичным сертификатом, полученным от удаленного SOA, передается объекту, которому требуется привилегия (то есть держателю привилегий).

6.3 Услугоцентричная модель

Управление доступом для телебиометрических применений является услугоцентричным в том смысле, что субъект доступа может вызывать множество различных функций, называемых *услугами доступа*. Услуга доступа – это некоторая функция, предоставляемая поставщиком услуг, который может оказать эту конкретную услуг. В качестве примера такой услуги можно назвать возможность выполнения различных манипуляций с данными пациентов в больнице. Указанием на конкретный тип услуги доступа служит идентификатор объекта. В настоящей Рекомендации не определяются конкретные услуги доступа, а предоставляются инструменты для организации оказания таких услуг. Услуги доступа могут определяться в ссылочных спецификациях сообразно сфере их применения.

Если у субъекта доступа есть доступ к определенной услуге, это не обязательно значит, что его привилегии распространяются на все аспекты этой услуги.

6.4 Объектно-атрибутная модель

Информационная модель, определенная в [ITU-T X.501], имеет целью охватить различные структуры данных. В настоящей Рекомендации эта модель используется в отношении управления доступом.

В модели [ITU-T X.501] защищаемые объекты организуются в классы по общим характеристикам, представляющим интерес в определенном контексте. Класс объекта обозначается идентификатором объекта. В [ITU-T X.1080.1] эта концепция получает дальнейшее развитие – идентификаторы объектов присваиваются группам классов объектов, которые соответствуют категориям, представляющим интерес в контексте телебиометрии.

В [ITU-T X.521] определяется набор общепримени мых классов объектов. Если возникает необходимость в классе объектов, не относящемся непосредственно к телебиометрии, во всех возможных случаях следует пользоваться уже определенным классом объектов.

Отдельные объекты обозначаются выделенными именами согласно определению в [ITU-T X.501]. Выделенное имя состоит из одного или нескольких компонентов, отражающих положение объекта в иерархии объектов.

Информация, связанная с объектом, моделируется в виде набора атрибутов. Атрибуты подразделяются на типы по общности характеристик. Тип атрибута обозначается идентификатором объекта. Атрибут – это последовательность идентификатора объекта, в которой указывается тип атрибута и одно или несколько значений этого типа. Объект может иметь только один атрибут конкретного типа. В [ITU-T X.520] определяется набор общепримени мых типов атрибутов. Во всех возможных случаях следует пользоваться уже определенными типами атрибутов. При необходимости в отдельных спецификациях могут быть определены дополнительные типы атрибутов.

При использовании настоящей Рекомендации в каждом конкретном случае необходимо установить соответствие между этой информационной моделью и фактической структурой базы данных. Это легко сделать, если информация хранится в справочнике (каталоге) LDAP или защищенном справочнике на основе спецификаций [ITU-T X.500-series].

6.5 Принципы базового управления доступом

Цель этого пункта – дать обзор общих принципов организации управления доступом для телебиометрии. В традиционных системах управления доступом учитывается главным образом право на доступ к данным или отказ в доступе к ним исходя из некоторых постоянных параметров. В настоящей Рекомендации принят более широкий подход, учитывающий также аспекты необходимости в запрашиваемых данных. Это достигается путем применения услугоцентричного подхода, рассматриваемого в пункте 6.3. Для выполнения той или иной операции субъект доступа должен

обладать привилегией на доступ к соответствующей услуге доступа, а также привилегией на доступ к необходимой информации.

Эта привилегия обеспечивает доступ к объектам одного или нескольких классов, возможно, ограниченный некоторым множеством именованных объектов. Тип разрешенной операции может различаться для разных классов объектов и именованных объектов.

Операции с объектами могут включать операции с отдельными атрибутами и их значениями. Разрешенные операции могут отличаться для разных типов атрибутов.

6.6 Взаимосвязь с другими схемами управления доступом

Существуют разные типы управления доступом в зависимости от различных требований к нему. В этом пункте кратко описываются различные схемы управления доступом и характеризуется их взаимосвязь с ACT.

6.6.1 Базовое управление доступом согласно ITU-T X.501

Базовое управление доступом согласно [ITU-T X.501] используется для защиты информации в справочнике, как определено в [ITU-T X.500-series]. Эта схема предусматривает подробные списки управления доступом, которые детально определяют разрешенные объекты и способы доступа для различных пользователей. Данная схема отличается от приведенной в настоящей Рекомендации тем, что учитывает только право на информацию, но не необходимость в ней.

6.6.2 Управление доступом на основе правил

Схема управления доступом на основе правил определяется как в [ITU-T X.501], так и в [b-ITU-T X.841]. При таком типе управления доступом подлежащие защите данные снабжаются меткой с информацией о том, какая защита требуется, а в запросе на доступ от пользователя указываются удостоверенные сведения об уровне допуска этого пользователя к определенной информации. Эта схема отличается от приведенной в настоящей Рекомендации тем, что требует помечать хранимые элементы данных, а также учитывает только право на информацию, но не необходимость в ней.

6.6.3 Управление доступом на основе ролей для "умных" сетей

В схеме управления доступом на основе ролей, описанной в [b-IEC 62351-8], акцент делается на пользователях и задачах, которые они выполняют. Роль – это совокупность прав доступа на объекты (действия, которые могут выполняться над определенными объектами). Пользователь может иметь одну или несколько ролей. Применительно к управлению доступом роль представляет собой своего рода промежуточный элемент, который сокращает требуемый объем информации в списке управления доступом (ACL) за счет снижения уровня ее детализации. Разрешения на доступ к объектам системы не указываются для каждого пользователя по отдельности; вместо этого пользователям назначаются роли, и права каждой роли описываются лишь один раз.

6.6.4 Управление доступом на основе атрибутов

Управление доступом на основе атрибутов (ABAC) – это модель логического управления доступом, которая отличается тем, что управление доступом к объектам осуществляется путем проверки выполнения правил, установленных в отношении атрибутов субъекта и объекта, операций и среды применительно к рассматриваемому запросу. Системы ABAC дают возможность реализовывать концепции как избирательного, так и обязательного управления доступом. ABAC обеспечивает точное управление доступом с учетом большего числа дискретных входных переменных при принятии решений, что позволяет оперировать более обширным множеством потенциальных комбинаций этих переменных и с их помощью отражать более широкий и полный набор возможных правил для выражения политик доступа.

Полезным введением в ABAC может послужить [b-NIST 800-162].

6.7 Обзор протоколов

6.7.1 Протокол заявления привилегий

Протокол заявления привилегий состоит из набора типов контента синтаксиса криптографическиъх сообщений (CMS). Для каждого типа доступа требуется соответствующий тип контента запроса и тип контента результата.

Экземпляр типа контента передается с использованием CMS согласно профилю, приведенному в Приложении B. Формальная спецификация протокола в виде модуля ASN.1 дается в Приложении C.

6.7.2 Протокол присваивания привилегий

Протокол присваивания привилегий служит для передачи сертификатов атрибутов между различными SOA или от SOA к держателю привилегий.

Для этого протокола определена одна пара типов контента – один для передачи информации о привилегиях в форме сертификатов атрибутов, другой – для извещения о результате.

Экземпляр типа контента передается с использованием CMS согласно профилю, приведенному в Приложении B. Формальная спецификация протокола в виде модуля ASN.1 дается в Приложении C.

6.8 Использование CMS

Профиль CMS для телебиометрических применений приведен в Приложении B.

Чтобы обеспечить надлежащую аутентификацию источника информации, контент тех типов, которые определены в настоящей Рекомендации, инкапсулируется в экземпляре типа контента **`signedData`**. Ввиду того что возможна передача конфиденциальной информации, рекомендуется дополнительно инкапсулировать его в экземпляре типа контента **`envelopedData`**. Тип контента **`ct-autEnvelopedData`** в настоящей Рекомендации не используется.

6.9 Соображения относительно сертификатов открытых ключей

Орган, выдавший сертификат атрибутов, подписывает его своим закрытым ключом. Такой сертификат должен верифицироваться с использованием соответствующего сертификата открытого ключа, выданного этому органу.

В принципе тем же закрытым ключом можно было бы подписать CMS-сообщение с контентом типа **`signedData`**, что облегчило бы процесс верификации. Однако использовать один и тот же закрытый ключ для разных целей может быть небезопасно. Следует рассмотреть возможность использования различных закрытых ключей для указанных двух целей, хотя это и не является обязательным требованием настоящей Рекомендации.

7 Передача информации о привилегиях

7.1 Использование сертификатов атрибутов

[ITU-T X.509] допускает использование как сертификатов открытых ключей, так и сертификатов атрибутов для передачи информации о привилегиях. В обоих случаях привилегии указываются в атрибутах, как определено в [ITU-T X.501]. Типы атрибутов, используемые для этой цели, отличаются от тех, с помощью которых моделируется информация об объектах. Типы атрибутов, предназначенные для передачи информации об объектах, должны быть как можно более простыми, тогда как типы атрибутов, несущие информацию о привилегиях, по самой своей природе являются довольно сложными.

В случае передачи информации о привилегиях в сертификате атрибутов:

a) компонент **`holder`** идентифицирует объект/субъект, которому присваиваются привилегии. Когда держатель привилегий предъявляет верификатору привилегий сертификат атрибутов с информацией о привилегиях, данный верификатор аутентифицирует субъекта доступа, чтобы убедиться, что тот действительно является держателем привилегий, указанным в сертификате;

b) компонент **`issuer`** содержит имя источника полномочий (SOA) или органа атрибутов (AA), которому делегированы полномочия присваивать привилегии. Кроме того, верификатор

привилегий получает и проверяет сертификат открытого ключа органа, выдавшего сертификат атрибутов, для проверки подписи на сертификате атрибутов;

c) компонент **attributes** содержит атрибут типа **accessService**, как определено в пункте 7.3.

Сертификат атрибутов подписывается SOA, утвердившим привилегию, или AA, которому делегированы полномочия по выдаче сертификатов.

Проверка упростится, если держатель привилегии и присвоивший ее орган имеют сертификаты открытого ключа, выданные одним и тем же органом по сертификации (CA).

7.2 Использование сертификатов открытых ключей

В случае передачи информации о привилегиях в сертификате открытого ключа:

a) компонент **subject** идентифицирует субъекта доступа, которому присвоены привилегии. Когда субъект доступа предъявляет этот сертификат открытого ключа верификатору привилегий, верификатор аутентифицирует субъекта доступа;

b) компонент **issuer** содержит выделенное имя CA, ответственного за выдачу данного сертификата открытого ключа;

c) расширение **subjectDirectoryAttributes** содержит экземпляр типа атрибута **accessService** (см. пункт 7.3).

7.3 Тип атрибута accessService

7.3.1 Синтаксис атрибута accessService

Атрибут типа **accessService** предназначен для включения в компонент **attributes** сертификата атрибутов либо в расширение **subjectDirectoryAttributes** сертификата открытого ключа. Он имеет следующий синтаксис:

```
accessService ATTRIBUTE ::= {
  WITH SYNTAX  AccessService
  ID           id-at-accessService }
```

Атрибут типа **AccessService** содержит информацию о привилегиях, которая позволяет верификатору привилегий принять решение о том, следует ли удовлетворить запрос на доступ или нет. Это многозначный тип атрибута, в котором можно перечислять несколько типов услуг доступа вместе с соответствующими разрешениями.

Субъект не может воспользоваться услугой, которая не указана в этом атрибуте.

Тип атрибута **accessService** имеет следующий синтаксис:

```
AccessService ::= SEQUENCE {
  serviceId          OBJECT IDENTIFIER,
  objectDef          SEQUENCE SIZE (1..MAX) OF ObjectSel,
  ... }
```

Значение типа данных **AccessService** имеет следующие компоненты:

a) компонент **serviceId** идентифицирует тип услуги доступа, в отношении которой у субъекта доступа есть некоторая привилегия;

b) в компоненте **objectDef** указываются классы объектов, в отношении которых держателю сертификата атрибутов для данного типа услуги присвоены привилегии. Он содержит по одному элементу на каждый класс объектов, в отношении которого субъекту присвоена привилегия. В этой услуге доступа держатель привилегий не имеет доступа к тем классам объектов, которые не перечислены в компоненте **objectDef**.

7.3.2 Выбор объектов

Выбор объектов, в отношении которых субъект доступа имеет привилегии, задается экземпляром типа данных **ObjectSel**:

```
ObjectSel ::= SEQUENCE {
```

```
  objecClass          OBJECT-CLASS.&id,
  objSelect           CHOICE {
    allObj         [0]  TargetSelect,
    objectNames    [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      object              CHOICE {
        names          [1]  SEQUENCE SIZE (1..MAX) OF DistinguishedName,
        subtree        [2]  DistinguishedName,
        ... },
      select               TargetSelect,
      ... },
    ... },
  ... }
```

Значение типа данных **ObjectSel** задает соответствующую привилегию для каждого класса объектов, в отношении которого присвоена привилегия для типа услуги, к которой осуществляется доступ. Этот тип данных имеет следующие компоненты:

– компонент **objectClass** задает класс объектов, в отношении которого присвоены привилегии;

– компонент **objSelect** задает объекты указанного класса, на которые распространяется присвоенная привилегия. У него есть две альтернативы:

  a) **allObj** выбирается, если присвоенные привилегии распространяются на все объекты класса. Содержит экземпляр типа данных **TargetSelect**;

  b) **objectNames** выбирается, если привилегии распространяются только на избранные объекты указанного класса. Может состоять из нескольких элементов, каждый из которых имеет следующие компоненты:

    i) компонент **object** задает один или несколько объектов, в отношении которых присвоены привилегии. У него есть следующие альтернативы:

      - **names,** которая задает имя одного или нескольких объектов, на которые распространяются привилегии;
      - **subtree,** которая используется для группы объектов, в которой выделенное имя или начальные компоненты имени каждого объекта совпадают с выделенным именем этой альтернативы;

    ii) компонент **select** содержит экземпляр типа данных **TargetSelect.**

Тип данных **TargetSelect** имеет следующий синтаксис:

```
TargetSelect ::= SEQUENCE {
  objOper   ObjectOperations OPTIONAL,
  attrSel   AttributeSel     OPTIONAL,
  ... }
  (WITH COMPONENTS {..., objOper  PRESENT } |
   WITH COMPONENTS {..., attrSel  PRESENT } )
```

Тип данных **TargetSelect** имеет следующие два необязательных компонента, из которых должен присутствовать хотя бы один:

a) компонент **objOper**, если он присутствует, задает допустимые операции над всеми или избранными объектами указанного класса. Если этот компонент отсутствует, никакие операции над объектами в целом не разрешены;

b) компонент **attrSel**, если он присутствует, содержит значение типа данных **AttributeSel** (см. пункт 7.3.3). Если этот компонент отсутствует, никакие операции над атрибутами объектов не разрешены.

7.3.3 Выбор типов атрибутов

```
AttributeSel ::= SEQUENCE {
  attSelect           CHOICE {
    allAttr        [0]  SEQUENCE {
      attrOper1         [0]  AttributeOperations OPTIONAL,
      ... },
```

```
    attributes      [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      select               SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
      attrOper2       [0]  AttributeOperations OPTIONAL,
      ... },
    ... },
  ... }
```

Тип данных **AttributeSel** задает типы атрибутов, на которые распространяется привилегия. У него есть следующий компонент:

– компонент **attSelect** имеет две альтернативы:

  a) **allAttr** выбирается, если привилегия распространяется на все атрибуты объекта(ов). У этой альтернативы есть следующий компонент:

    i) компонент **attrOper1** задает допустимые операции над атрибутами;

  b) **attributes** выбирается, если привилегия распространяется только на некоторые атрибуты объекта(ов). Субъект доступа не имеет привилегий в отношении неперечисленных типов атрибутов и не уведомляется о них. Эта альтернатива состоит из следующих компонентов:

    i) компонент **select** задает один или несколько типов атрибутов, на которые распространяются привилегии;

    ii) компонент **attrOper2** задает операции, которые могут выполняться над атрибутами.

7.4 Операции над объектами в целом

Допустимые операции над каким-либо объектом задаются с помощью следующего типа данных:

```
ObjectOperations ::= BIT STRING {
  read            (0),
  add             (1),
  modify          (2),
  delete          (3),
  rename          (4),
  discloseOnError (5) }
```

Разрешение **read** устанавливается, если субъекту доступа разрешается считывать информацию из объекта.

Разрешение **add** устанавливается, если субъекту доступа разрешается добавлять новые объекты конкретного класса. Оно требует разрешения **add** на все объекты соответствующего класса. Для каждого атрибута, добавляемого к объекту, разрешение **add** выдается на соответствующий тип атрибута (см. пункт 7.5).

Разрешение **modify** устанавливается, если субъекту доступа разрешается изменять существующий объект. У субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы), подлежащий(е) изменению. Если субъект доступа добавляет атрибуты, у него должно быть разрешение **add** на соответствующие типы атрибутов. Если субъект доступа удаляет атрибуты, у него должно быть разрешение **delete** на соответствующие типы атрибутов. Если субъект доступа изменяет атрибуты, у него должно быть разрешение **modify** на соответствующие типы атрибутов. Если субъект доступа удаляет значение(я) атрибутов, у него должно быть разрешение **deleteValue** на соответствующие типы атрибутов. Если субъект доступа заменяет атрибуты, у него должно быть разрешение **replaceAttribute** на соответствующие типы атрибутов.

Разрешение **delete** устанавливается, если субъекту доступа разрешается удалять существующий объект. У субъекта доступа должно быть разрешение **delete** на класс объектов в целом или на поименованный(е) объект(ы), подлежащий(е) удалению.

Разрешение **rename** устанавливается, если субъекту доступа разрешается переименовывать существующий(е) объект(ы). У субъекта доступа должно быть разрешение **rename** на класс объектов в целом или на поименованный(е) объект(ы), подлежащий(е) переименованию.

Разрешение **discloseOnError** устанавливается, если субъекту доступа разрешается знать о существовании объекта в случае, когда выполнить операцию не удается.

7.5 Операции над атрибутами

```
AttributeOperations ::= BIT STRING {
  read              (0),
  compare           (1),
  add               (2),
  modify            (3),
  delete            (4),
  deleteValue       (5),
  replaceAttribute  (6),
  discloseOnError   (7) }
```

Разрешение **read** устанавливается для каждого типа атрибутов, которые разрешается считывать субъекту доступа. У субъекта доступа должно быть разрешение **read** на соответствующий класс объектов в целом или на соответствующие поименованные объекты. Кроме того, он должен иметь разрешение **read** на все атрибуты этих классов объектов или доступ к соответствующему(им) типу(ам) атрибутов.

Разрешение **compare** устанавливается, если субъекту доступа разрешается сравнивать один или несколько атрибутов. У субъекта доступа должно быть разрешение **read** на класс объектов в целом или на поименованный(е) объект(ы). Кроме того, у него должно быть разрешение **compare** на все атрибуты разрешенных классов объектов или соответствующий(е) тип(ы) атрибутов.

Разрешение **add** устанавливается, если субъекту доступа разрешается добавлять один или несколько атрибутов. У субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы). Кроме того, у него должно быть разрешение **add** на соответствующие типы атрибутов.

Разрешение **modify** устанавливается, если субъекту доступа разрешается изменять атрибут конкретного типа. Кроме того, у субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы).

Разрешение **delete** устанавливается, если субъекту доступа разрешается удалять один или несколько атрибутов. Кроме того, у субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы).

Разрешение **deleteValue** устанавливается, если субъекту доступа разрешается удалять одно или несколько значений из атрибутов указанного(ых) типа(ов). Кроме того, у субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы).

Разрешение **replaceAttribute** устанавливается, если субъекту доступа разрешается заменять атрибут данного типа другим атрибутом того же типа. Кроме того, у субъекта доступа должно быть разрешение **modify** на класс объектов в целом или на поименованный(е) объект(ы).

Разрешение **discloseOnError** устанавливается, если субъекту доступа разрешается знать о существовании атрибута в случае, когда выполнить операцию не удается. Кроме того, у субъекта доступа должно быть разрешение **discloseOnError** на объект в целом.

7.6 Обработка ошибок

При использовании CMS могут возникать ошибки (см. Приложение A.5). Если обнаружена ошибка, необходимость в дальнейшей проверке отсутствует. В качестве результата запроса на доступ возвращается ошибка CMS.

Кроме того, ошибки могут возникать и по итогам самого запроса на доступ.

Об ошибке сообщается в экземпляре типа данных **AccessdErr**, определение которого дано в пункт 8.10.

8 Протокол заявления привилегий

8.1 Общий обзор

Следующая совокупность информационных объектов включает в себя все типы контента, представленные информационными объектами, которые определены в настоящей Рекомендации.

```
ActContentTypes CONTENT-TYPE ::= {
    privAssignRequest |
    privAssignResult |
    readRequest |
    readResult |
    compareRequest |
    compareResult |
    addRequest |
    addResult |
    deleteRequest |
    deleteResult |
    modifyRequest |
    modifyResult |
    renameRequest |
    renameResult,
    ... }
```

Типы контента, заданные совокупностью **ActContentTypes**, образуют протокол заявления привилегий, который предусматривает ряд различных операций доступа, описанных в пунктах 8.4–8.9.

8.2 Общие компоненты запросов

Все запросы включают следующие компоненты:

```
CommonReqComp ::= SEQUENCE {
  attrCerts [31]  AttributeCertificates OPTIONAL,
  serviceId [30]  OBJECT IDENTIFIER,
  invokId   [29]  INTEGER,
  ... }

AtributeCertificates ::= SEQUENCE SIZE (1..MAX) OF AttributeCertificate
```

Общие параметры запросов таковы.

a) Компонент **attrCert**, если он присутствует, задает сертификат атрибутов или тракт сертификации атрибутов, содержащий привилегию субъекта доступа. Если этот компонент отсутствует, информация о привилегии для субъекта доступа передается в сертификате открытого ключа оконечного объекта.

b) Компонент **serviceId** задает тип запрашиваемой услуги.

c) Компонент **invokId** принимает нулевое значение для первой запрошенной операции и затем увеличивается на единицу для каждой последующей запрашиваемой операции. Диапазон его значений должен быть таким, чтобы одно и то же значение не использовалось повторно на протяжении значительного периода времени при взаимодействии между двумя объектами. Он предоставляется для обнаружения пропущенных запросов и атак с повторной передачей. Получатель запроса указывает в своем ответе то же значение, чтобы субъект доступа мог определить, какому запросу соответствует полученный результат.

8.3 Доступ к услуге

Все типы операций, перечисленные в пунктах 8.4–8.9, требуют доступа к конкретной услуге. Верификатор привилегий (получатель) проверяет, что привилегия, присвоенная субъекту доступа в соответствующем сертификате атрибутов или сертификате открытого ключа, действительно разрешает доступ к запрошенной услуге.

Если у субъекта доступа нет разрешения на использование запрошенного типа услуги или поставщик услуг не поддерживает этот тип услуги, возвращается код ошибки **noSuchService**.

Если у субъекта доступа есть разрешение на использование услуги, проверяется, соответствует ли запрошенная операция данному типу услуги, и в случае несоответствия возвращается код ошибки **invalidOperationForService**.

8.4 Операция чтения

Операция чтения состоит из запроса на чтение и его соответствующего результата.

Запрос на чтение передается как экземпляр типа контента **readRequest**, а результат – как экземпляр типа контента **readResult**:

```
readRequest CONTENT-TYPE ::= {
              ReadRequest
IDENTIFIED BY id-readRequest
```

Субъект доступа использует тип контента **readRequest** для считывания (чтения) информации о конкретном объекте:

```
ReadRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  selection [2] InformationSelection,
  ... }
```

Тип данных **ReadRequest** задает синтаксис фактического контента и имеет следующие компоненты:

a) компонент **object** содержит выделенное имя объекта, о котором запрашивается информация;

b) компонент **selection** задает тип информации, запрашиваемой субъектом доступа (см. пункт 8.11).

Запрос на чтение не выполняется, если в запросе указан неизвестный объект. В этом случае возвращается код ошибки **noSuchObject.**

Запрос на чтение не выполняется, если у субъекта доступа отсутствует разрешение **read** на объект в соответствии с присвоенными ему привилегиями. Если разрешение **read** отсутствует, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа обладает разрешением **discloseOnError** на этот объект, в противном случае возвращается код ошибки **noSuchObject.**

Разрешение **read** на соответствующий тип атрибута требуется для каждого атрибута, который должен быть возвращен в результате запроса. Если у субъекта доступа отсутствует разрешение **read** на конкретный тип атрибута, атрибут этого типа не возвращается. Если в итоге никакие атрибуты не подлежат возврату, запрос не выполняется. Если у субъекта доступа есть разрешение **discloseOnError** на все атрибуты, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth,** в противном случае возвращается код ошибки **noInformation**:

```
readResult CONTENT-TYPE ::= {
              ReadResult
IDENTIFIED BY id-readResult }
```

Верификатор привилегий использует экземпляр типа контента **readResult** для возврата запрошенной информации или для сообщения об ошибке:

```
ReadResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }
```

Тип данных **ReadResult** задает синтаксис фактического контента и имеет следующие компоненты:

a) компонент **object** содержит имя объекта, о котором запрашивается информация;

b) компонент **result** содержит результат запроса на чтение. У него есть две альтернативы:

   – **success** выбирается, если есть информация, подлежащая возврату. Содержит экземпляр типа данных **ObjectInformation** (см. пункт 8.12). Возвращаемая информация представляет собой пересечение между той информацией, которую запросил субъект доступа, и той, которую ему разрешено получить;

– **failure** выбирается, если предстоит сообщить об ошибке.

8.5 Операция сравнения

Операция сравнения состоит из запроса на сравнение и его соответствующего результата.

Запрос на сравнение передается как экземпляр типа контента **compareRequest**, а результат – как экземпляр типа контента **compareResult**:

```
compareRequest CONTENT-TYPE ::= {
               CompareRequest
IDENTIFIED BY id-compareRequest }
```

Экземпляр типа контента **compareRequest** используется для сравнения переданного в запросе предполагаемого значения атрибута конкретного типа со значением атрибута того же типа, принадлежащего конкретному объекту:

```
CompareRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  purported [2] AttributeValueAssertion,
  ... }
```

Тип данных **CompareRequest** задает фактический контент и имеет следующие компоненты, помимо тех, которые определены в пункте 8.2:

a) компонент **object** содержит выделенное имя объекта, значение атрибута которого подлежит сравнению;

b) компонент **purported** содержит комбинацию типа и значения атрибута, которая подлежит сравнению с однотипным атрибутом рассматриваемого объекта.

Запрос на сравнение не выполняется, если в запросе указан неизвестный объект. В этом случае возвращается код ошибки **noSuchObject.**

Запрос на сравнение не выполняется, если у субъекта доступа отсутствует разрешение **read** на объект в соответствии с присвоенными ему привилегиями. Если разрешение **read** на объект отсутствует, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа обладает разрешением **discloseOnError** на этот объект, в противном случае возвращается код ошибки **noSuchObject.**

Запрос на сравнение не выполняется, если у субъекта доступа отсутствует разрешение **compare** на соответствующий тип атрибута. Если у субъекта доступа есть разрешение **discloseOnError** на этот тип атрибута, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth,** в противном случае возвращается код ошибки **noInformation**:

```
compareResult CONTENT-TYPE ::= {
              CompareResult
IDENTIFIED BY id-compareResult
```

Верификатор привилегий использует экземпляр типа контента **compareResult** для возврата запрошенной информации или для сообщения об ошибке:

```
CompareResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] CompareOK,
    failure   [1] AccessdErr,
    ... },
  ... }

CompareOK ::= SEQUENCE {
  matched        [0] BOOLEAN,
  matchedSubtype [1] BOOLEAN OPTIONAL,
  ... }
```

Тип данных **CompareResult** задает синтаксис фактического контента и имеет следующие компоненты:

a) компонент **object** содержит выделенное имя объекта, в отношении которого поступил запрос на сравнение;

b) компонент **result** содержит результат запроса на сравнение. У него есть две альтернативы:

– **success** выбирается, если есть информация, подлежащая возврату. Возвращается экземпляр типа данных **CompareOK** со следующими компонентами:

i) компонент **matched**, которому присваивается значение **TRUE**, если значение атрибута указанного типа или одного из его подтипов равняется тому значению, которое передано в запросе. В случае совпадения по подтипу должен также присутствовать компонент **matchedSubtype** со значением **TRUE**. Компоненту **matched** присваивается значение **FALSE**, если совпадение по указанному типу или его подтипам отсутствует;

– **failure** выбирается, если предстоит сообщить об ошибке.

8.6 Операция добавления

Операция добавления состоит из запроса на добавление и его соответствующего результата.

Запрос на добавление передается как экземпляр типа контента **addRequest**, а результат – как экземпляр типа контента **addResult**:

```
addRequest CONTENT-TYPE ::= {
              AddRequest
IDENTIFIED BY id-addRequest }
```

Для добавления нового объекта в информационную систему используется экземпляр типа контента **addRequest**:

```
AddRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  attr      [2] SEQUENCE SIZE (1..MAX) OF Attribute {{SupportedAttributes}}
                OPTIONAL,
  ... }
```

Тип данных **AddResult** задает синтаксис фактического контента и имеет следующие компоненты:

a) компонент **object** содержит выделенное имя добавляемого объекта;

b) компонент **attr**, если он присутствует, содержит один или несколько атрибутов, которые должны быть приданы новому объекту.

Если у субъекта доступа отсутствует разрешение **add** на указанный класс объекта, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth.**

Если объект с таким выделенным именем уже существует и при этом у субъекта доступа есть разрешение **discloseOnError**, возвращается код ошибки **objectAlreadyExists**, в противном случае возвращается код ошибки **insufficientAccessRigth.**

Если у субъекта доступа отсутствует разрешение **add** на все атрибуты, которые должны быть приданы объекту, запрос не выполняется. Если у субъекта доступа есть разрешение **discloseOnError** на все перечисленные типы атрибутов, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth**, в противном случае возвращается код ошибки **noInformation**:

```
addResult CONTENT-TYPE ::= {
              AddResult
IDENTIFIED BY id-addResult }
```

Верификатор привилегий использует экземпляр типа контента **addResult** для возврата запрошенной информации или для сообщения об ошибке:

```
AddResult ::= CHOICE {
  success    [0] NULL,
  failure    [1] AccessdErr,
  ... }
```

Тип данных **AddResult** задает синтаксис фактического контента и имеет следующие компоненты:

a) **success** выбирается, если объект был добавлен;

b) **failure** выбирается, если предстоит возвратить сообщение об ошибке.

8.7 Операция удаления

Операция удаления состоит из запроса на удаление и его соответствующего результата.

Запрос на удаление передается как экземпляр типа контента **deleteRequest**, а результат – как экземпляр типа контента **deleteResult**:

```
deleteRequest CONTENT-TYPE ::= {
              DeleteRequest
IDENTIFIED BY id-deleteRequest }
```

Для удаления существующего объекта из информационной системы используется экземпляр типа контента **deleteRequest**:

```
DeleteRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  ... }
```

Тип данных **DeleteRequest** задает синтаксис фактического контента и имеет следующие компоненты.

a) Компонент **object** содержит выделенное имя удаляемого объекта.

Если запрошено удаление несуществующего объекта, возвращается код ошибки **noSuchObject.**

Если у субъекта доступа отсутствует разрешение **delete** на класс объекта, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа имеет разрешение **discloseOnError** на этот объект, в противном случае возвращается код ошибки **noSuchObject**:

```
deleteResult CONTENT-TYPE ::= {
              DeleteResult
IDENTIFIED BY id-deleteResult }
```

Верификатор привилегий использует экземпляр типа контента **deleteResult** для возврата запрошенной информации или для сообщения об ошибке:

```
DeleteResult ::= CHOICE {
  success    [0] NULL,
  failure    [1] AccessdErr,
  ... }
```

Экземпляр типа данных **DeleteResult** имеет две альтернативы:

a) **success** выбирается, если объект был удален;

b) **failure** выбирается, если предстоит сообщить об ошибке.

8.8 Операция изменения

Операция изменения состоит из запроса на изменение и его соответствующего результата.

Запрос на изменение передается как экземпляр типа контента **modifyRequest**, а результат – как экземпляр типа контента **modifyResult**:

```
modifyRequest CONTENT-TYPE ::= {
              ModifyRequest
IDENTIFIED BY id-modifyRequest }
```

Для изменения существующего объекта используется экземпляр типа контента **modifyRequest**:

```
ModifyRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  changes       SEQUENCE SIZE (1..MAX) OF ObjectModification,
  select        InformationSelection,
  ... }

ObjectModification ::= CHOICE {
  addAttribute     [0]  Attribute{{SupportedAttributes}},
  deleteAttribute  [1]  AttributeType,
  addValues        [2]  Attribute{{SupportedAttributes}},
  deleteValues     [3]  Attribute{{SupportedAttributes}},
  replaceAttribute [4]  Attribute{{SupportedAttributes}},
  ... }
```

Тип данных **ModifyResult** задает синтаксис фактического контента и имеет следующие компоненты.

a) Компонент **object** содержит выделенное имя изменяемого объекта:

- если такого объекта не существует, возвращается код ошибки **noSuchObject**;
- если у субъекта доступа отсутствует разрешение **modify** на объект, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа имеет разрешение **discloseOnError** на этот объект, в противном случае возвращается код ошибки **noSuchObject**.

b) Компонент **changes** содержит информацию об изменениях в одном или нескольких атрибутах. Он имеет следующий набор альтернатив.

- Альтернатива **addAttribute** содержит новый атрибут, который требуется добавить:
  - i) если у субъекта доступа отсутствует разрешение **add** на указанный тип атрибута, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth**;
  - ii) если атрибут такого типа уже существует, возвращается код ошибки **attributeAlreadyExists** при условии, что у субъекта доступа есть разрешение **discloseOnError** на этот тип атрибута, в противном случае возвращается код ошибки **insufficientAccessRigth**.
- Альтернатива **deleteAttribute** идентифицирует атрибут, который требуется удалить:
  - i) если у субъекта доступа отсутствует разрешение **delete** на указанный тип атрибута, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth**;
  - ii) если атрибут такого типа не существует, возвращается код ошибки **noSuchAttribute**.
- Альтернатива **addValues** идентифицирует существующий атрибут по его типу. В ней указываются значения, которые требуется добавить к атрибуту:
  - i) если у объекта нет атрибута указанного типа, возвращается код ошибки **noSuchAttribute** при условии, что субъект доступа имеет разрешение **discloseOnError**, в противном случае возвращается код ошибки **insufficientAccessRigth**;
  - ii) если у субъекта доступа отсутствует разрешение **addValue**, в описанной выше ситуации возвращается код ошибки **insufficientAccessRigth**;
  - iii) в случае попытки добавить уже имеющееся значение возвращается код ошибки **attributeValueAlreadyExists** при условии, что у субъекта доступа есть разрешение **discloseOnError**, в противном случае возвращается код ошибки **insufficientAccessRigth**.
- Альтернатива **deleteValues** идентифицирует атрибут по его типу. В ней указываются значения, которые требуется удалить из атрибута:
  - i) если у объекта нет атрибута указанного типа, возвращается код ошибки **noSuchAttribute** при условии, что субъект доступа имеет разрешение

discloseOnError, в противном случае возвращается код ошибки **insufficientAccessRigth**;

– если у субъекта доступа отсутствует разрешение **deleteValue**, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа имеет разрешение **discloseOnError**, в противном случае возвращается код ошибки **noSuchAttributeValue**;

iii) при попытке удалить несуществующее значение атрибута возвращается код ошибки **noSuchAttributeValue**.

– Альтернатива **replaceAttribute** используется для замены существующего атрибута новым атрибутом того же типа:

i) если у объекта нет атрибута указанного типа, возвращается код ошибки **noSuchAttribute** при условии, что субъект доступа имеет разрешение **discloseOnError**, в противном случае возвращается код ошибки **insufficientAccessRigth**;

– если у субъекта доступа отсутствует разрешение **replaceAttribute**, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа имеет разрешение **discloseOnError**, в противном случае возвращается код ошибки **noSuchAttribute**:

```
modifyResult CONTENT-TYPE ::= {
             ModifyResult
IDENTIFIED BY id-modifyResult }
```

Верификатор привилегий использует экземпляр типа контента **modifyResult** для возврата запрошенной информации или для сообщения об ошибке:

```
ModifyResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }
```

Экземпляр типа данных **ModifyResult** имеет две альтернативы:

a) **success** выбирается, если объект был изменен;

b) **failure** выбирается, если предстоит возвратить сообщение об ошибке.

8.9 Операция переименования объекта

Операция переименования состоит из запроса на переименование и его соответствующего результата.

Запрос на переименование передается как экземпляр типа контента **renameRequest**, а результат – как экземпляр типа контента **renameResult**:

```
renameRequest CONTENT-TYPE ::= {
              RenameRequest
IDENTIFIED BY id-renameRequest }
```

Для переименования существующего объекта используется экземпляр типа контента **renameRequest**:

```
RenameRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  new           DistinguishedName,
  ... }
```

Тип данных **RenameRequest** задает синтаксис фактического контента и имеет следующие компоненты:

a) в компоненте **object** указывается существующее на данный момент выделенное имя объекта, который необходимо переименовать;

b) компонент **new** содержит новое выделенное имя объекта.

Если запрошено переименование несуществующего объекта, возвращается код ошибки **noSuchObject.**

Если у субъекта доступа отсутствует разрешение **rename** на поименованный объект, возвращается код ошибки **insufficientAccessRigth** при условии, что субъект доступа имеет разрешение **discloseOnError** на этот объект, в противном случае возвращается код ошибки **noSuchObject**:

```
renameResult CONTENT-TYPE ::= {
              RenameResult
IDENTIFIED BY id-renameResult }

RenameResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] NULL,
    failure   [1] AccessdErr,
    ... },
  ... }
```

Экземпляр типа данных **RenameResult** имеет две альтернативы:

a) **success** выбирается, если объект был изменен;

b) **failure** выбирается, если предстоит возвратить сообщение об ошибке.

8.10 Обработка ошибок

Если в ходе обработки запроса возникает исключительная ситуация, получатель должен возвратить код ошибки, включив в результат экземпляр типа данных **AccessErr**:

```
AccessdErr ::= CHOICE {
  cmsErr     [0] CmsErr,
  ActErr   [1] PbactErr,
  ... }
```

Альтернатива **cmsErr** выбирается в случае, если исключительная ситуация возникла при оценке типов контента, определенных посредством CMS (см. п. A.5).

Альтернатива **pbactErr** выбирается в случае, если оценка экземпляров типов контента CMS прошла без ошибок, но обнаружена ошибка в инкапсулированном экземпляре типа контента, относящегося к PBACT.

8.11 Выбор информации

Тип данных **InformationSelection** используется для определения того, какая информация затребована в запросе на чтение или изменение:

```
InformationSelection ::= SEQUENCE {
  attributes      CHOICE {
    allAttributes   [0]  NULL,
    select          [1]  SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
    ... },
  infoTypes        ENUMERATED {
    attributeTypesOnly     (0),
    attributeTypeAndValues (1),
    ... },
  ...}
```

Тип данных **InformationSelection** имеет следующие компоненты:

a) компонент **attributes** задает возвращаемые атрибуты. У него есть две альтернативы:

- **allAttributes** выбирается, если субъект доступа запрашивает всю информацию об объекте;
- **select** выбирается, если запрошен только избранный набор атрибутов;

b) компонент **infoTypes** представляет собой перечисление со следующими элементами:

– элемент **attributeTypesOnly** выбирается, если возврату подлежат только типы атрибутов. В этом случае у субъекта доступа должно быть разрешение read на соответствующий тип атрибута согласно текущим привилегиям. Если это не так, соответствующий тип атрибута исключается из возвращаемого результата. Если в итоге не остается информации, подлежащей возврату, запрос не выполняется;

– элемент **attributeTypesAndValues** выбирается, если согласно действующим привилегиям возврату подлежат как типы, так и значения атрибутов. В этом случае у субъекта доступа должно быть разрешение read на соответствующий тип атрибута согласно текущим привилегиям. Если это не так, соответствующие тип и значения атрибута исключаются из возвращаемого результата. Если в итоге не остается информации, подлежащей возврату, запрос не выполняется.

8.12 Информация об объекте

Если есть информация об объекте, подлежащая возврату, она возвращается в виде экземпляра следующего типа данных:

```
ObjectInformation ::= SEQUENCE {
  object  DistinguishedName,
  info    CHOICE {
    attr    SET SIZE (1..MAX) OF Attribute {{SupportedAttributes}},
    type    SET SIZE (1..MAX) OF AttributeType },
  ... }
```

Компонент **object** содержит выделенное имя объекта, информация о котором возвращается.

Компонент **info** содержит набор атрибутов, несущих запрошенную информацию, или набор типов атрибутов.

При отсутствии информации, подлежащей возврату, запрос не выполняется.

8.13 Определенные коды ошибок

Здесь определяются коды ошибок для конкретных типов контента PBACT:

```
PbactErr ::= ENUMERATED {
  noSuchService,
  invalidOperationForService,
  insufficientAccessRigth,
  noSuchObject,
  noSuchAttribute,
  noSuchAttributeValue,
  objectAlreadyExists,
  attributeAlreadyExists,
  attributeValueAlreadyExists,
  noInformation,
  ... }
```

a) Код ошибки **noSuchService** возвращается, если субъект доступа запрашивает услугу, на которую у него нет разрешения, о которой ему не разрешено знать или которая не поддерживается.

b) Код ошибки **invalidOperationForService** возвращается, если запрошенная операция не относится к запрошенной услуге.

c) Код ошибки **insufficientAccessRigth** возвращается, когда субъект доступа запрашивает услугу, на которую у него нет разрешения или когда он хочет совершить операцию, на которую у него нет разрешения в соответствии с услугой, к которой осуществляется доступ.

d) Код ошибки **noSuchObject** возвращается, если субъект доступа пытается получить доступ к несуществующему объекту или объекту, о существовании которого ему не разрешено знать.

e) Код ошибки **noSuchAttribute** возвращается, если субъект доступа пытается получить доступ к несуществующему атрибуту или атрибуту, о существовании которого ему не разрешено знать.

f) Код ошибки **noSuchAttributeValue** возвращается, если субъект доступа пытается получить доступ к несуществующему значению атрибута или к значению, о существовании которого ему не разрешено знать.

g) Код ошибки **objectAlreadyExists** возвращается при попытке добавить объект, выделенное имя которого совпадает с выделенным именем уже существующего объекта, если субъекту доступа разрешено знать о существовании этого объекта.

h) Код ошибки **attributeAlreadyExists** возвращается при попытке добавить атрибут типа к объекту, который уже содержит атрибут того же типа, если субъекту доступа разрешено знать о существовании этого типа атрибута в объекте.

i) Код ошибки **attributeValueAlreadyExists** возвращается при попытке добавить атрибут к объекту, который уже содержит атрибут того же типа, если субъекту доступа разрешено знать о существовании этого атрибута.

j) Код ошибки **noInformation** возвращается, если запрошенная информация недоступна или субъекту доступа не разрешено знать о существовании этих данных.

9 Протокол присваивания привилегий

9.1 Сфера применения протокола

Протокол присваивания привилегий используется для присваивания привилегий:

a) SOA промежуточному AA для дальнейшего делегирования;

b) источником SOA непосредственно некоторому субъекту, которому предстоит заявлять эти привилегии;

c) органом AA непосредственно некоторому субъекту, которому предстоит заявлять эти привилегии;

d) органом AA другому AA в ситуации, когда тракт делегирования от SOA держателю привилегий проходит через несколько AA.

ПРИМЕЧАНИЕ. – Рекомендуется делать тракт делегирования как можно более коротким.

9.2 Типы контента

В протоколе присваивания привилегий используются два типа контента: один собственно для присваивания привилегий, а другой – для подтверждения присвоения.

9.2.1 Тип контента запрос на присваивание привилегий

Запрос на присваивание привилегий передается в экземпляре типа контента **privAssignRequest**:

```
privAssignRequest CONTENT-TYPE ::= {
                PrivAssignRequest
IDENTIFIED BY id-privAssignRequest}
```

Синтаксис фактического контента задается следующим типом данных:

```
PrivAssignRequest ::= SEQUENCE {
  attrCerts   AttributeCertificates OPTIONAL,
  ... }
```

Этот тип данных имеет только один компонент, содержащий последовательность сертификатов атрибутов. Если запрос на присваивание привилегий поступает от SOA, последовательность состоит лишь из одного сертификата атрибутов. Если между SOA и держателем привилегий имеется только один промежуточный AA, в запрос от AA держателю привилегий включаются как сертификат атрибутов, выданный SOA, так и сертификат атрибутов, выданный AA. Еще один сертификат атрибутов требуется на каждый дополнительный AA в тракте делегирования между SOA и держателем привилегий.

9.2.2 Тип контента результат запроса на присваивание привилегий

Результат запроса на присваивание привилегий передается в экземпляре типа контента **privAssignResult**:

```
privAssignResult CONTENT-TYPE ::= {
              PrivAssignResult
IDENTIFIED BY id-privAssignResult }
```

Синтаксис фактического контента задается следующим типом данных:

```
PrivAssignResult ::= SEQUENCE {
  result  CHOICE {
    success  NULL,
    failure  PrivAssignErr },
  ... }

PrivAssignErr ::= CHOICE {
--cmsErr     [0] CmsErr,
  assignErr  [1] AssignErr,
  ... }
```

9.2.3 Коды ошибок

```
AssignErr ::= ENUMERATED {
  invalidAttributeCertificate (0),
  invalidDelegationPath
  invalidPublicKeyCertificate
    ... }
```

Приложение A

Распределение идентификаторов объектов, используемых в серии МСЭ-Т X.1080

(Данное Приложение является неотъемлемой частью настоящей Рекомендации)

A.1 Верхний уровень дерева идентификаторов объектов

В Приложении A к [ITU-T X.1081] дуги распределяются ниже дуги, выделенной для телебиометрии, то есть:

```
id-telebio OBJECT IDENFIER ::= { joint-iso-itu-t(2) telebiometrics(42) }
```

Ниже этой дуги в [ITU-T X.1081] следующая дуга выделяется для телездравоохранения:

```
id-th OBJECT IDENTIFIER ::= { id-telebio th(3) }
```

В [ITU-T X.1080.1] ниже дуги **id-th** распределено несколько дуг. Дуга 0 выделена для модулей ASN.1, которые определены в [ITU-T X.1080.1]. Другие дуги распределены категориям объектов. Настоящая Рекомендация устанавливает, что дуга 0 также используется для распределения идентификаторов объектов, используемых в Рекомендациях МСЭ-Т серии X.1080 в целом. Чтобы избежать конфликтов, для распределения идентификаторов объектов в контексте Рекомендаций МСЭ-Т серии X.1080 используется значение 10:

```
id-telehelth OBJECT IDENTIFIER ::= { id-th all(0) telehealth(10) }
```

Различным частям Рекомендаций МСЭ-Т серии X.1080 распределены следующие дуги:

```
id-x1080-0 OBJECT IDENTIFIER ::= { id-telehelth part0(0) }
id-x1080-1 OBJECT IDENTIFIER ::= { id-telehelth part1(1) }
id-x1080-2 OBJECT IDENTIFIER ::= { id-telehelth part2(2) }
----
```

Распределенная конкретной части Рекомендаций МСЭ-Т серии X.1080 дуга подразделяется следующим образом:

– для модулей выделена дуга 0;

– для типов CMS-контента выделена дуга 1;

– для типов атрибутов выделена дуга 2.

В конкретных частях по необходимости могут выделяться дополнительные дуги.

В контексте настоящей Рекомендации для модулей выделена следующая дуга:

```
id-x1080-0-module OBJECT IDENTIFIER ::= { id-x1080-0 module(0) }
```

Для типов CMS-контента выделена следующая дуга:

```
id-x1080-0-Cont   OBJECT IDENTIFIER ::= { id-x1080-0 cmsCont(1) }
```

Для типов атрибутов, используемых для присваивания привилегий, выделена следующая дуга:

```
id- x1080-0-attr   OBJECT IDENTIFIER ::= { id-x1080-0 prAttr(2) }
```

A.2 Идентификаторы объектов для типов контента CMS

Следующие идентификаторы объектов распределены типам контента, определенным для протокола присваивания привилегий и протокола заявления привилегий:

```
id-privAssignReq  OBJECT IDENTIFIER ::= { id-x1080-0-Cont privAssignRequest(1) }
id-privAssignRes  OBJECT IDENTIFIER ::= { id-x1080-0-Cont privAssignResult(2) }
id-readRequest    OBJECT IDENTIFIER ::= { id-x1080-0-Cont readRequest(3) }
id-readResult     OBJECT IDENTIFIER ::= { id-x1080-0-Cont readResult(4) }
id-compareRequest OBJECT IDENTIFIER ::= { id-x1080-0-Cont compareRequest(5) }
```

```
id-compareResult  OBJECT IDENTIFIER ::= { id-x1080-0-Cont compareResult(6) }
id-addRequest     OBJECT IDENTIFIER ::= { id-x1080-0-Cont addRequest(7) }
id-addResult      OBJECT IDENTIFIER ::= { id-x1080-0-Cont addResult(8) }
id-deleteRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont deleteRequest(9) }
id-deleteResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont deleteResult(10) }
id-modifyRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont modifyRequest(11) }
id-modifyResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont modifyResult(12) }
id-renameRequest  OBJECT IDENTIFIER ::= { id-x1080-0-Cont renameRequest(13) }
id-renameResult   OBJECT IDENTIFIER ::= { id-x1080-0-Cont renameResult(14) }
```

A.3 Идентификаторы объектов для типов атрибутов, относящихся к привилегиям

```
id-at-accessSer   OBJECT IDENTIFIER ::= { id-pbactPrivAttr 1 }
```

Приложение B

Профиль синтаксиса криптографических сообщений

(Данное Приложение является неотъемлемой частью настоящей Рекомендации)

B.1 Общие сведения

Синтаксис криптографических сообщений (CMS) определен в [IETF RFC 5652]. Он определяет возможности связи, позволяющие обеспечивать целостность, аутентификацию и конфиденциальность данных. В [IETF RFC 5083] приведены дополнительные спецификации. В [IETF RFC 5911] и [IETF RFC 6268] содержатся новые модули ASN.1 для CMS. В настоящем Приложении дается профиль CMS для использования в телебиометрических спецификациях со ссылками на вышеуказанные спецификации.

CMS представляет собой универсальную спецификацию, предназначенную для использования во многих различных контекстах. В описываемом здесь профиле задействованы не все возможности CMS. Этот профиль включает типы данных CMS, используемые в настоящей Рекомендации и других телебиометрических спецификациях. Другие телебиометрические спецификации могут ссылаться на данное Приложение при использовании CMS.

В настоящем Приложении нет намерения представить спецификацию реализации, которая не соответствует спецификациям IETF RFC. Предполагается лишь обсудить аспекты CMS, относящиеся к телебиометрическим спецификациям. В Добавлении I содержится неформальный модуль ASN.1, отражающий использование CMS в телебиометрии.

В CMS определены различные типы контента, каждый из которых имеет свое назначение. В телебиометрических спецификациях используются следующие типы контента: **signedData** (для аутентификации и обеспечения целостности данных), **envelopedData** (для шифрования и тем самым обеспечения конфиденциальности) и **ct-authEnvelopedData**. Тип контента **signedData** используется, когда требуется цифровая подпись, а **envelopedData** – когда требуется конфиденциальность. В случае использования типа контента **envelopedData** его экземпляр инкапсулируется в экземпляре типа контента **signedData**. Тип контента **ct-authEnvelopedData** используется, когда множество сообщений образуют конкретную задачу.

Не все аспекты упомянутых выше типов контента используются в телебиометрических спецификациях. Поэтому в настоящем Приложении приводится профиль для использования CMS в телебиометрии. Для облегчения поиска информации здесь изложены представляющие интерес аспекты CMS.

Экземпляр типа контента, определенного в телебиометрических спецификациях, инкапсулируется в экземпляре типа контента **envelopedData**, если требуется конфиденциальность, а в противном случае – в экземпляре типа контента **signedData**. Как вариант, он может включаться в экземпляр типа контента **ct-authEnvelopedData**.

Тип контента согласно [IETF RFC 6268] определяется с использованием следующего класса информационного объекта:

```
CONTENT-TYPE ::= TYPE-IDENTIFIER
```

Класс информационного объекта **CONTENT-TYPE** эквивалентен встроенному классу информационного объекта ASN.1 **TYPE-IDENTIFIER**. Информационный объект **CONTENT-TYPE** используется для привязки типа контента, обозначенного идентификатором объекта, к абстрактному синтаксису контента.

Тип данных **ContentInfo**, определение которого приведено ниже, задает общий синтаксис типа контента:

```
ContentInfo ::= SEQUENCE {
  contentType  CONTENT-TYPE.&id ({TelebSupportedcontentTypes}),
  content      CONTENT-TYPE.&Type
         ({TelebSupportedcontentTypes}{@contentType})OPTIONAL,
  ... }
```

```
TelebSupportedcontentTypes CONTENT-TYPE ::=
  { signedData | envelopedData | ct-authEnvelopedData, ...}
```

К поддерживаемым типам контента относятся такие типы контента, как **signedData**, **envelopedData**, **ct-authEnvelopedData**, а также набор типов контента, определенных в конкретной телебиометрической спецификации.

CMS требует указывать версию CMS для типа данных, чтобы определить конкретный синтаксис, который используется с этим типом. Определены следующие версии:

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

В [IETF RFC 6268] определен следующий параметризованный тип данных, используемых во всех спецификациях:

```
Attributes { ATTRIBUTE:AttrList } ::=
             SET SIZE (1..MAX) OF Attribute {{ AttrList }}
```

B.2 Использование типа контента signedData

Указанный ниже тип контента определен в пункте 5 [IETF RFC 5652]. В слегка измененной нотации, отражающей его использование для целей телебиометрии, он определяется следующим образом:

```
signedData CONTENT-TYPE ::= {
                SignedData
  IDENTIFIED BY id-signedData }

SignedData ::= SEQUENCE  {
  version               CMSVersion (v3),
  digestAlgorithms      SET (SIZE (1)) OF AlgorithmIdentifier
                          {{Teleb-Hash-Algorithms}},
  encapContentInfo      EncapsulatedContentInfo,
  certificates     [0]  IMPLICIT SET (SIZE (1..MAX)) OF Certificate OPTIONAL,
  crls             [1]  IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos           SignerInfos,
  ... }
```

ПРИМЕЧАНИЕ 1. – [IETF RFC 6268] оперирует несколько измененной версией типа данных **AlgorithmIdentifier**. Однако в настоящем профиле используется тип данных **AlgorithmIdentifier,** как он определен в [ITU-T X.509].

Компонент **version** принимает значение **v3** в соответствии с пунктом 5.1 [IETF RFC 5652].

Компонент **digestAlgorithms** состоит из одного элемента, задающего алгоритм хеширования из набора допустимых алгоритмов в соответствии с применимой телебиометрической спецификацией.

Следующее определение допускает указание любого алгоритма хеширования:

```
Teleb-Hash-Algorithms ALGORITHM ::= {...}
```

ПРИМЕЧАНИЕ 2. – Настоящий профиль не предписывает использования в обязательном порядке какого-либо конкретного алгоритма хеширования. В ссылочных спецификациях или соглашениях, заключаемых с пользователями Рекомендаций, многоточия могут заменяться набором алгоритмов хеширования, которые должны поддерживаться в определенном контексте.

ПРИМЕЧАНИЕ 3. – CMS допускает использование нескольких цифровых подписей, а следовательно, и применение нескольких алгоритмов хеширования. Случай использования нескольких цифровых подписей не представляет интереса в контексте телебиометрических спецификаций.

Компонент **encapContentInfo** содержит экземпляр следующего типа данных:

```
EncapsulatedContentInfo ::= SEQUENCE {
  eContentType     CONTENT-TYPE.&id({envelopedData, ...}),
  eContent     [0] EXPLICIT OCTET STRING
    (CONTAINING CONTENT-TYPE.&Type({envelopedData, ...}
     {@eContentType})) OPTIONAL }
```

Этот тип данных имеет следующие компоненты:

a) компонент **eContentType** содержит идентификатор объекта, указывающий на тип инкапсулированного контента. Если требуется шифрование, этот компонент содержит идентификатор типа контента **envelopedData**. Если шифрование не требуется, он содержит один из типов контента, определенных в соответствующей телебиометрической спецификации;

b) компонент **econtent** содержит фактический инкапсулированный контент с оберткой в виде строки октетов. Он присутствует всегда.

ПРИМЕЧАНИЕ 4. – Этот компонент определен как необязательный. Однакоо для всех типов контента, о которых идет речь, определен соответствующий контент.

Компонент **certificates** содержит сертификаты открытых ключей, достаточные для установления единого тракта сертификации, представляемого типом данных **PkiPath**, который определен в [ITU-T X.509].

Компонент **crls** не имеет отношения к телебиометрическим спецификациям и должен отсутствовать.

Компонент **signerInfos** содержит экземпляр типа данных **SignerInfos**:

```
SignerInfos ::= SET (SIZE (1)) OF SignerInfo

SignerInfo ::= SEQUENCE {
  version                 CMSVersion (v1),
  sid                     SignerIdentifier,
  digestAlgorithm         AlgorithmIdentifier {{Teleb-Hash-Algorithms}},
  signedAttrs        [0]  IMPLICIT Attributes{{SignedAttributes}} OPTIONAL,
  signatureAlgorithm      AlgorithmIdentifier {{Teleb-Signature-Algorithms}},
  signature               SignatureValue,
  unsignedAttrs      [1]  IMPLICIT Attributes {{UnsignedAttributes}} OPTIONAL,
  ... }

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

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

SignedAttributes ATTRIBUTE ::= { contentType | messageDigest, ... }

Teleb-Signature-Algorithms ALGORITHM ::= {...}

SignatureValue ::= OCTET STRING

UnsignedAttributes ATTRIBUTE ::= {...}
```

Данный профиль поддерживает лишь случай, когда существует одна подписавшая сторона, поэтому тип данных **SignerInfos** имеет один и только один элемент. Тип данных **SignerInfo** имеет следующие компоненты:

a) компонент **version** принимает значение **v1** в соответствии с [IETF RFC 5652];

b) компонент **sid** идентифицирует сертификат открытого ключа оконечной сущности подписавшей стороны и содержит экземпляр типа данных **SignerIdentifier**. Этот тип данных задает две альтернативы:

   – альтернатива **issuerAndSerialNumber** идентифицирует сертификат открытого ключа оконечного объекта по выделенному имени CA, выдавшего сертификат, и серийному номеру сертификата. Эта альтернатива выбирается во всех случаях;

   – альтернатива **subjectKeyIdentifier** не выбирается;

c) компонент **digestAlgorithm** принимает такое же значение, как и указанное в компоненте **digestAlgorithms** типа данных **SignerInfo**;

d) компонент **signedAttrs** содержит список подписанных атрибутов. [IETF RFC 5652] требует включения в этот компонент по крайней мере экземпляров типов атрибутов **contentType** и **messageDigest**. Настоящий профиль не требует включения в список каких-либо дополнительных атрибутов, но в ссылочных спецификациях могут быть дополнения к этому списку;

e) компонент **signatureAlgorithm** содержит алгоритм цифровой подписи, который использовался для создания цифровой подписи, содержащейся в компоненте **signature;**

ПРИМЕЧАНИЕ 5. – Настоящий профиль не предписывает использования в обязательном порядке какого-либо конкретного алгоритма цифровой подписи. В ссылочных спецификациях или соглашениях, заключаемых с пользователями Рекомендаций, многоточия могут заменяться набором алгоритмов цифровой подписи, которые должны поддерживаться в определенном контексте.

f) компоненты **signature** и **unsignedAttrs** соответствуют [IETF RFC 5652].

B.3 Использование типа контента envelopedData

B.3.1 Общие сведения

Тип контента **envelopedData** обеспечивает возможность шифрования данных. Для этого требуется создать набор общих симметричных ключей. В [IETF RFC 5652] предусматриваются различные методы генерации таких симметричных ключей. Настоящий профиль требует применения метода соглашения о ключах, известного как метод соглашения о ключах Диффи-Хелмана. Этот метод описан в [IETF RFC 2631] для случая без использования эллиптических кривых. В [IETF RFC 5753] даны спецификации для использования методов на базе эллиптических кривых.

В результате применения метода DH получают общий секретный ключ, который может быть использован в качестве исходного материала для генерации общих симметричных ключей. В настоящем профиле предусматриваются два режима работы по методу DH – эфемерно-статический и статико-статический.

Эфемерно-статический режим требует, чтобы у получателя был сертификат открытого ключа DH, выданный органом по сертификации. Этот сертификат открытого ключа должен быть доступен отправителю. Отправитель создает новую пару ключей DH для каждого отправляемого сообщения. Таким образом каждое сообщение имеет отдельный общий секретный ключ.

Статико-статический режим требует, чтобы у каждого из взаимодействующих объектов был сертификат открытого ключа DH, выданный органом по сертификации. Поскольку применение этого режима может привести к использованию одного и того же общего секретного ключа для всех сообщений, отправитель должен представлять случайный материал ключа пользователя, чтобы обеспечить разный материал для генерации ключа для каждого сообщения.

Настоящий профиль требует поддержки эфемерно-статического режима.

Оба метода требуют, чтобы у обоих взаимодействующих объектов был принадлежащий партнеру сертификат открытого DH-ключа оконечного объекта, выданный органом по сертификации, поскольку взаимодействие является двусторонним.

Указанный ниже тип контента определен в пункте 6 [IETF RFC 5652] с корректировками в [IETF RFC 6268]:

```
envelopedData CONTENT-TYPE ::= {
                EnvelopedData
  IDENTIFIED BY id-envelopedData }

EnvelopedData ::= SEQUENCE {
  version                   CMSVersion(v0 | v2),
  originatorInfo         [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  encryptedContentInfo      EncryptedContentInfo,
  ...,
```

```
  [[2: unprotectedAttrs  [1] IMPLICIT Attributes
       {{UnprotectedAttributes}} OPTIONAL ]] }

UnprotectedAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }
```

Компонент **EnvelopedData** имеет следующие компоненты:

a) компонент **version** согласно [IETF RFC 5652] принимает значение **v2**, если присутствует компонент **unprotectedAttrs**. В противном случае он принимает значение **v0**;

b) компонент **originatorInfo** отсутствует;

c) компонент **recipientInfos** содержит экземпляр типа данных **RecipientInfos**, определенный в пункте B.3.2;

d) компонент **encryptedContentInfo** содержит экземпляр типа данных **EncryptedContentInfo**, как определено в пункте B.3.4;

e) компонент **unprotectedAttrs** обязателен, если ожидается, что следующим в соответствующем направлении будет передаваться экземпляр типа контента **ct-authEnvelopedData**, в противном случае он может отсутствовать.

B.3.2 Информация о получателе

Тип данных **RecipientInfos** допускает использование нескольких экземпляров типа данных **RecipientInfo**. Однако для целей настоящего профиля количество экземпляров ограничивается одним. Экземпляр типа данных **RecipientInfo** задает альтернативные способы получения общего секретного ключа двумя взаимодействующими сторонами:

```
RecipientInfos ::= SET SIZE (1) OF RecipientInfo

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

В настоящем профиле используются только две альтернативы для типа данных **RecipientInfo**, определенные в [IETF RFC 5652]:

a) альтернатива **kari** – единственный вариант, разрешенный для использования с типом контента **envelopedData**. Тип данных **KeyAgreeRecipientInfo** предоставляет информацию, необходимую для получения общего секретного ключа, как описано в пункте B.3.3;

a) альтернатива **kekri** – единственный вариант, разрешенный для использования с типом контента **ct-authEnvelopedData**. Тип данных **KEKRecipientInfo** предоставляет информацию, необходимую для получения общего секретного ключа, как описано в пункте B.4.

B.3.3 Соглашение о ключах

```
KeyAgreeRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v3),
  originator         [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm                [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys,
  ... }

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

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

SupportedDHPublicKeyAlgorithms ALGORITHM ::= {...}

UserKeyingMaterial ::= OCTET STRING (SIZE (64))

KeyEncryptionAlgorithmIdentifier ::=
    AlgorithmIdentifier{{SupportedKeyIncryptAlgorithms}}

SupportedKeyIncryptAlgorithms ALGORITHM ::= {...}

RecipientEncryptedKeys ::= SEQUENCE (SIZE (1)) OF RecipientEncryptedKey

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

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

EncryptedKey ::= OCTET STRING
```

Тип данных **KeyAgreeRecipientInfo** имеет следующие компоненты.

a) Компонент **version** согласно [IETF RFC 5652] принимает значение **v3**.

b) Компонент **originator** содержит экземпляр типа данных **OriginatorIdentifierOrKey** со следующими альтернативами:

- альтернатива **issuerAndSerialNumber** выбирается, если используется статико-статический метод. Она содержит экземпляр типа данных **IssuerAndSerialNumber**. Этот тип данных идентифицирует принадлежащий отправителю сертификат открытого ключа DH;
  - i) компонент **issuer** содержит выделенное имя CA, выдавшего сертификат, и устанавливается идентичным компоненту **issuer** соответствующим сертификатом открытого ключа;
  - ii) компонент **serialNumber** устанавливается идентичным компоненту **serialNumber** соответствующим сертификатом открытого ключа;
- альтернатива **subjectKeyIdentifier** не выбирается;
- альтернатива **originatorKey** выбирается в случае использования эфемерно-статического метода DH. Она содержит экземпляр типа данных **OriginatorPublicKey** со следующими компонентами;
  - i) компонент **algorithm** содержит ссылку на использованный алгоритм генерации открытого ключа DH;

ПРИМЕЧАНИЕ 1. – Настоящий профиль не предписывает использования в обязательном порядке какого-либо конкретного алгоритма генерации открытого ключа DH. В ссылочных спецификациях или соглашениях, заключаемых с пользователями Рекомендаций, многоточия могут заменяться набором алгоритмов генерации открытого ключа DH, которые должны поддерживаться в определенном контексте.

  - ii) компонент **publicKey** содержит открытый ключ DH, генерированный получателем. Отправитель генерирует пару ключей по методу DH для каждого использования этого типа контента.

На основе локального закрытого ключа и открытого ключа получателя отправитель может генерировать общий секретный ключ. Получатель генерирует идентичный общий секретный ключ, используя свой закрытый ключ вместе с открытым ключом отправителя, содержащимся в типе данных **OriginatorPublicKey** или **IssuerAndSerialNumber**.

c) Компонент **ukm** присутствует в случае использования статико-статического метода и содержит экземпляр типа данных **UserKeyingMaterial.**

На основе общего секретного ключа, значения компонента **ukm** (если он используется) и некоторой другой информации, определенной в [IETF RFC 2631], обе стороны генерируют так называемый ключ шифрования ключей (KEK). Этот ключ затем используется для шифрования генерированного отправителем ключа шифрования контента (CEK). Данный метод называют обертыванием ключа.

d) Компонент **keyEncryptionAlgorithm** задает алгоритм обертывания ключа и содержит экземпляр типа данных **KeyEncryptionAlgorithmIdentifier.**

ПРИМЕЧАНИЕ 2. – Настоящий профиль не предписывает использования в обязательном порядке какого-либо конкретного набора алгоритмов обертывания ключа. В будущем могут быть определены новые алгоритмы. Алгоритмы обертывания ключа усовершенствованного стандарта шифрования (AES) определены в [IETF RFC 3394]. В ссылочных спецификациях или соглашениях, заключаемых с пользователями Рекомендаций, многоточия могут заменяться набором алгоритмов обертывания ключа, который должен поддерживаться в определенном контексте.

e) Компонент **recipientEncryptedKeys** содержит экземпляр типа данных **RecipientEncryptedKeys**. Такой экземпляр должен состоять из одного элемента, то есть одного экземпляра типа данных **RecipientEncryptedKey**. Этот тип данных имеет следующие два компонента:

- компонент **rid** идентифицирует получателя по его сертификату открытого ключа оконечного объекта;
- компонент **encryptedKey** содержит шифрованный CEK, используемый для шифрования контента, как описано в подпункте c).

B.3.4 Многократное использование ключей шифрования контента CMS

Если CEK используется для шифрования последующего экземпляра типа контента **ct-authEnvelopedData**, как определено в [IETF RFC 3185], в незащищенные атрибуты вводится соответствующая исходная информация, как описано в пункте B.3.1. Эта информация сохраняется обеими сторонами. Если требуется высокий уровень безопасности, атрибут типа **aa-CEKMaxDecrypts** должен иметь значение 1 или должен быть опущен.

B.3.5 Шифрованный контент

Экземпляр типа данных **EncryptedContentInfo** содержит шифрованный инкапсулированный контент:

```
EncryptedContentInfo ::= SEQUENCE {
  contentType                 CONTENT-TYPE.&id ({EncryptedContentSet}),
  contentEncryptionAlgorithm SEQUENCE {
    algorithm                   ALGORITHM.&id ({SymmetricEncryptionAlgorithms}),
    parameter                   ALGORITHM.&Type
            ({SymmetricEncryptionAlgorithms}{@.algorithm})} OPTIONAL,
  encryptedContent       [0] IMPLICIT EncryptedContent OPTIONAL,
  ... }

EncryptedContentSet CONTENT-TYPE ::= {...}

SymmetricEncryptionAlgorithms ALGORITHM ::= {...}

EncryptedContent ::= OCTET STRING
```

Компонент **encryptedContentInfo** типа данных **EnvelopedData** содержит экземпляр типа данных **EncryptedContentInfo**:

a) компонент **ContentType** содержит информацию о типе шифрованного контента. Перечень возможных типов контента включает те типы, для которых доступно шифрование;

b) компоненты **contentEncryptionAlgorithm** и **encryptedContent** соответствуют [IETF RFC 5652].

B.4 Использование типа контента ct-authEnvelopedData

B.4.1 Общие сведения

В соответствии с [ITU-T X.1080.1] структура протоколов для телебиометрических применений в общем случае такова: подготовительный обмен данными для установления сеанса, затем многократный обмен информацией, и, наконец, завершение сеанса. В таком контексте может отсутствовать необходимость в создании нового ключа шифрования ключей для каждого сообщения.

В [IETF RFC 5083] определяется тип контента **ct-authEnvelopedData**, не включенный в [IETF RFC 5652]. Этот тип контента позволяет использовать эффективные методы шифрования с аутентификацией. В настоящем профиле применяются алгоритмы AES-GCM, определенный в [IETF RFC 5084], наряду с многократным использованием CEK согласно [IETF RFC 3185]. Для получения подробной информации см. соответствующие спецификации.

Тип контента **ct-authEnvelopedData** определяется следующим образом:

```
ct-authEnvelopedData CONTENT-TYPE ::= {
                AuthEnvelopedData
  IDENTIFIED BY id-ct-authEnvelopedData }

AuthEnvelopedData ::= SEQUENCE {
  version                   CMSVersion (v0),
  originatorInfo       [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  authEncryptedContentInfo EncryptedContentInfo,
  authAttrs            [1] IMPLICIT Attributes {{AuthAttributes}} OPTIONAL,
  mac                       MessageAuthenticationCode,
  unauthAttrs          [2] IMPLICIT Attributes {{UnauthAttributes}} OPTIONAL }

AuthAttributes ATTRIBUTE ::= {...}

MessageAuthenticationCode ::= OCTET STRING

UnauthAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }
```

Тип данных **AuthEnvelopedData** имеет следующие компоненты:

a) компонент **version** согласно [IETF RFC 5083] принимает значение **v0**;

b) компонент **originatorInfo** отсутствует;

c) компонент **recipientInfos** содержит экземпляр типа данных **RecipientInfos**. Этот тип данных описан в пункте B.3.2. Для данного типа контента возможен выбор альтернативы **kekri**, помимо альтернативы **kari**. Если выбирается альтернатива **kekri**, она содержит экземпляр типа данных **KEKRecipientInfo**, как определено в пункте B.4.2;

d) компонент **authEncryptedContentInfo** содержит экземпляр типа данных **EncryptedContentInfo**, как определено в пункте B.3.5;

e) компонент **authAttrs**, если он присутствует, содержит набор атрибутов, которые подлежат защите посредством аутентификации;

f) компонент **mac** содержит генерированный код аутентификации сообщений (MAC);

g) компонент **unauthAttrs** содержит атрибуты того же типа, как определено в пункте B.3.1, подпункт e). Если известно, что этот экземпляр типа контента будет последним в данном сеансе в текущем направлении, то этот компонент может быть опущен.

B.4.2 Информация о получателе KEK

```
KEKRecipientInfo ::= SEQUENCE {
  version                CMSVersion (v4),
  kekid                  KEKIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey           EncryptedKey }

KEKIdentifier ::= SEQUENCE {
```

```
  keyIdentifier OCTET STRING,
  date          GeneralizedTime OPTIONAL,
  other         OtherKeyAttribute OPTIONAL,
  ... }
```

Тип данных **KEKRecipientInfo** имеет следующие компоненты:

a) компонент **version** согласно [IETF RFC 5652] принимает значение **v4**;

b) компонент **kekid** содержит экземпляр типа данных **KEKIdentifier** со следующими компонентами:

   – компонент **keyIdentifier** содержит идентификатор CEK, сохраненный от предыдущего обмена данными, как определено в пункте B.3.4;

   – прочие компоненты необязательны;

c) компонент **keyEncryptionAlgorithm** содержит алгоритм обертывания ключа, как определено в пункте B.3.3, подпункт d). Рекомендуется использовать один и тот же алгоритм обертывания ключа для всех экземпляров контента в конкретном телебиометрическом сеансе.

B.5 Атрибуты

Приведенные ниже типы атрибутов определены в [IETF RFC 5652]. Экземпляры этих типов атрибутов предназначены для использования в качестве подписанных атрибутов:

```
contentType ATTRIBUTE ::= {
  WITH SYNTAX            CONTENT-TYPE.&id({envelopedData, ...})
  EQUALITY MATCHING RULE objectIdentifierMatch
  SINGLE VALUE           TRUE
  ID                     id-contentType }

messageDigest ATTRIBUTE ::= {
  WITH SYNTAX            OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-messageDigest }
```

Приведенные ниже типы атрибутов определены в [IETF RFC 3185]. Экземпляры этих типов атрибутов предназначены для использования в качестве неподписанных атрибутов:

```
aa-CEKReference ATTRIBUTE ::= {
  WITH SYNTAX            CEKReference
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE           TRUE
  ID                     id-aa-CEKReference }

CEKReference ::= OCTET STRING

aa-CEKMaxDecrypts ATTRIBUTE ::= {
  WITH SYNTAX            CEKMaxDecrypts
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE           TRUE
  ID                     id-aa-CEKMaxDecrypts }

CEKMaxDecrypts ::= INTEGER

aa-KEKDerivationAlg ATTRIBUTE ::= {
  WITH SYNTAX            KEKDerivationAlgorithm
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE           TRUE
  ID                     id-aa-KEKDerivationAlg }

KEKDerivationAlgorithm ::= SEQUENCE {
  kekAlg       AlgorithmIdentifier,
  pbkdf2Param  PBKDF2-params }
```

```
PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
--  otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}}
    ... },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}},
  ... }

PBKDF2-PRFs ALGORITHM ::= {...}

PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
    otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}} },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}} DEFAULT algid-hmacWithSHA1
}


id-pkcs OBJECT IDENTIFIER ::=
   { iso(1) member-body(2) usa(840) rsadsi(113549) pkcs(1) }


id-pkcs-9 OBJECT IDENTIFIER ::= { id-pkcs pkcs-9(9) }

id-aa OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) attributes(2) }

id-contentType         OBJECT IDENTIFIER ::= { id-pkcs-9 3 }
id-messageDigest       OBJECT IDENTIFIER ::= { id-pkcs-9 4 }
id-aa-CEKReference     OBJECT IDENTIFIER ::= { id aa 30 }
id-aa-CEKMaxDecrypts   OBJECT IDENTIFIER ::= { id aa 31 }
id-aa-KEKDerivationAlg OBJECT IDENTIFIER ::= { id aa 32 }
```

B.6 Коды ошибок синтаксиса криптографических сообщений

[b-IETF RFC 7191] содержит список кодов ошибок CMS для всех возможных вариантов использования CMS. Ниже представлен поднабор этих кодов ошибок, относящийся к телебиометрии. Описание кодов ошибок см. в [b-IETF RFC 7191].

Когда в [b-IETF RFC 7191] дается ссылка на сертификат, имеется в виду сертификат открытого ключа, используемый для типов контента, определенных с использованием CMS.

```
CmsErrorCode ::= ENUMERATED {
  decodeFailure                    (1),
  badContentInfo                   (2),
  badSignedData                    (3),
  badEncapContent                  (4),
  badCertificate                   (5),
  badSignerInfo                    (6),
  badSignedAttrs                   (7),
  badUnsignedAttrs                 (8),
  missingContent                   (9),
  noTrustAnchor                   (10),
  notAuthorized                   (11),
  badDigestAlgorithm              (12),
  badSignatureAlgorithm           (13),
  unsupportedKeySize              (14),
  unsupportedParameters           (15),
  signatureFailure                (16),
  incorrectTarget                 (23),
  missingSignature                (29),
```

```
versionNumberMismatch            (31),
revokedCertificate               (33),
badEncryptedData                 (62),
badEnvelopedData                 (63),
badKeyAgreeRecipientInfo         (66),
badKEKRecipientInfo              (67),
badEncryptContent                (68),
badEncryptAlgorithm              (69),
missingCiphertext                (70),
decryptFailure                   (71),
badMACAlgorithm                  (72),
badAuthAttrs                     (73),
badUnauthAttrs                   (74),
invalidMAC                       (75),
mismatchedDigestAlg              (76),
missingCertificate               (77),
tooManySigners                   (78),
missingSignedAttributes          (79),
derEncodingNotUsed               (80),
invalidAttributeLocation         (82),
badAttributes                    (85),
noMatchingRecipientInfo          (91),
unsupportedKeyWrapAlgorithm      (92),
badKeyTransRecipientInfo         (93),
other                           (127) }
```

Приложение C

Формальная спецификация протоколов заявления и присваивания привилегий

(Данное Приложение является неотъемлемой частью настоящей Рекомендации)

```
Pbact-access { joint-iso-itu-t(2) telebiometrics(42) e-health-protocol(3)
   modules(0) pbact-access(6) version1(1) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- EXPORTS All

IMPORTS

  -- из Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2

  ATTRIBUTE, Attribute{}, AttributeType, AttributeTypeAndValue,
AttributeValueAssertion, DistinguishedName, OBJECT-CLASS, SupportedAttributes
    FROM InformationFramework {joint-iso-itu-t ds(5) module(1)
informationFramework(1) 8}

  -- из Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8

  AttributeCertificate
    FROM AttributeCertificateDefinitions {joint-iso-itu-t ds(5) module(1)
         attributeCertificateDefinitions(32) 8}

  CmsErrorCode, CONTENT-TYPE
    FROM CmsTelebiometric { joint-iso-itu-t(2) telebiometrics(42) th(3) part0(0)
modules(0) cmsProfile(1) version1(1) } ;

accessService ATTRIBUTE ::= {
  WITH SYNTAX  AccessService
  ID           id-at-accessService }

AccessService ::= SEQUENCE {
  serviceId          OBJECT IDENTIFIER,
  objectDef          SEQUENCE SIZE (1..MAX) OF ObjectSel,
  ... }

ObjectSel ::= SEQUENCE {
  objecClass         OBJECT-CLASS.&id,
  objSelect          CHOICE {
    allObj        [0]  TargetSelect,
    objectNames   [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      object             CHOICE {
        names         [1]  SEQUENCE SIZE (1..MAX) OF DistinguishedName,
        subtree       [2]  DistinguishedName,
        ... },
      select           TargetSelect,
      ... },
    ... },
  ... }

TargetSelect ::= SEQUENCE {
  objOper   ObjectOperations OPTIONAL,
  attrSel   AttributeSel     OPTIONAL,
  ... }
  (WITH COMPONENTS {..., objOper  PRESENT } |
   WITH COMPONENTS {..., attrSel  PRESENT } )
```

```
AttributeSel ::= SEQUENCE {
  attSelect           CHOICE {
    allAttr        [0]  SEQUENCE {
      attrOper1      [0]  AttributeOperations OPTIONAL,
      ... },
    attributes     [1]  SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      select              SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
      attrOper2      [0]  AttributeOperations OPTIONAL,
      ... },
    ... },
  ... }

ObjectOperations ::= BIT STRING {
  read            (0),
  add             (1),
  modify          (2),
  delete          (3),
  rename          (4),
  discloseOnError (5) }

AttributeOperations ::= BIT STRING {
  read             (0),
  compare          (1),
  add              (2),
  modify           (3),
  delete           (4),
  deleteValue      (5),
  replaceAttribute (6),
  discloseOnError  (7) }

PbactContentTypes CONTENT-TYPE ::= {
    privAssignRequest |
    privAssignResult |
    readRequest |
    readResult |
    compareRequest |
    compareResult |
    addRequest |
    addResult |
    deleteRequest |
    deleteResult |
    modifyRequest |
    modifyResult |
    renameRequest |
    renameResult,
    ... }

CommonReqComp ::= SEQUENCE {
  attrCerts [31]  AttributeCertificates OPTIONAL,
  serviceId [30]  OBJECT IDENTIFIER,
  invokId   [29]  INTEGER,
  ... }

AttributeCertificates ::= SEQUENCE SIZE (1..MAX) OF AttributeCertificate

readRequest CONTENT-TYPE ::= {
              ReadRequest
IDENTIFIED BY id-readRequest }

ReadRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  selection [2] InformationSelection,
  ... }
```

```
readResult CONTENT-TYPE ::= {
              ReadResult
IDENTIFIED BY id-readResult }


ReadResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }

compareRequest CONTENT-TYPE ::= {
              CompareRequest
IDENTIFIED BY id-compareRequest }

CompareRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  purported [2] AttributeValueAssertion,
  ... }

compareResult CONTENT-TYPE ::= {
              CompareResult
IDENTIFIED BY id-compareResult }

CompareResult ::= SEQUENCE {
  object    DistinguishedName,
  result    CHOICE {
    success   [0] CompareOK,
    failure   [1] AccessdErr,
    ... },
  ... }

CompareOK ::= SEQUENCE {
  matched        [0] BOOLEAN,
  matchedSubtype [1] BOOLEAN DEFAULT FALSE,
  ... }

addRequest CONTENT-TYPE ::= {
              AddRequest
IDENTIFIED BY id-addRequest }

AddRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object    [1] DistinguishedName,
  attr      [2] SEQUENCE SIZE (1..MAX) OF Attribute {{SupportedAttributes}}
                  OPTIONAL,
  ... }

addResult CONTENT-TYPE ::= {
              AddResult
IDENTIFIED BY id-addResult }

AddResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }

deleteRequest CONTENT-TYPE ::= {
              DeleteRequest
IDENTIFIED BY id-deleteRequest }
```

```
DeleteRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  ... }

deleteResult CONTENT-TYPE ::= {
              DeleteResult
IDENTIFIED BY id-deleteResult }

DeleteResult ::= CHOICE {
  success   [0] NULL,
  failure   [1] AccessdErr,
  ... }

modifyRequest CONTENT-TYPE ::= {
              ModifyRequest
IDENTIFIED BY id-modifyRequest }

ModifyRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  changes       SEQUENCE SIZE (1..MAX) OF ObjectModification,
  select        InformationSelection,
  ... }

ObjectModification ::= CHOICE {
  addAttribute     [0]  Attribute{{SupportedAttributes}},
  deleteAttribute  [1]  AttributeType,
  addValues        [2]  Attribute{{SupportedAttributes}},
  deleteValues     [3]  Attribute{{SupportedAttributes}},
  replaceAttribute [4]  Attribute{{SupportedAttributes}},
  ... }

modifyResult CONTENT-TYPE ::= {
              ModifyResult
IDENTIFIED BY id-modifyResult }

ModifyResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] ObjectInformation,
    failure   [1] AccessdErr,
    ... },
  ... }

renameRequest CONTENT-TYPE ::= {
              RenameRequest
IDENTIFIED BY id-renameRequest }

RenameRequest ::= SEQUENCE {
  COMPONENTS OF CommonReqComp,
  object        DistinguishedName,
  new           DistinguishedName,
  ... }

renameResult CONTENT-TYPE ::= {
              RenameResult
IDENTIFIED BY id-renameResult }

RenameResult ::= SEQUENCE {
  result    CHOICE {
    success   [0] NULL,
    failure   [1] AccessdErr,
    ... },
```

```
  ... }

AccessdErr ::= CHOICE {
  cmsErr     [0] CmsErrorCode,
  pbactErr   [1] PbactErr,
  ... }

InformationSelection ::= SEQUENCE {
  attributes      CHOICE {
    allAttributes   [0]  NULL,
    select          [1]  SEQUENCE SIZE (1..MAX) OF ATTRIBUTE.&id,
    ... },
  infoTypes        ENUMERATED {
    attributeTypesOnly     (0),
    attributeTypeAndValue  (1),
    ... },
  ... }

ObjectInformation ::= SEQUENCE {
  name   DistinguishedName,
  info   SET SIZE (1..MAX) OF Attribute {{SupportedAttributes}},
  ... }

PbactErr ::= ENUMERATED {
  noSuchService,
  invalidOperationForService,
  insufficientAccessRigth,
  noSuchObject,
  noSuchAttribute,
  noSuchAttributeValue,
  objectAlreadyExists,
  attributeAlreadyExists,
  attributeValueAlreadyExists,
  noInformation,
  ... }

privAssignRequest CONTENT-TYPE ::= {
              PrivAssignRequest
IDENTIFIED BY id-privAssignRequest }

PrivAssignRequest ::= SEQUENCE {
  attrCerts [1]  AttributeCertificates OPTIONAL,
  ... }

privAssignResult CONTENT-TYPE ::= {
              PrivAssignResult
IDENTIFIED BY id-privAssignResult }

PrivAssignResult ::= SEQUENCE {
  result  CHOICE {
    success  NULL,
    failure  PrivAssignErr },
  ... }

PrivAssignErr ::= CHOICE {
  cmsErr     [0] CmsErrorCode,
  assignErr  [1] AssignErr,
  ... }

AssignErr ::= ENUMERATED {
  invalidAttributeCertificate (0),
  ... }

-- распределение идентификаторов объектов
```

```
-- дерево верхнего уровня

id-pbact              OBJECT IDENTIFIER ::=
  { joint-iso-itu-t(2) telebiometrics(42) e-health-protocol(3) pbact(20) }
id-pbactmodule        OBJECT IDENTIFIER ::= { id-pbact module(0) }
id-pbactCont          OBJECT IDENTIFIER ::= { id-pbact cmsCont(1) }
id-pbactPrivAttr      OBJECT IDENTIFIER ::= { id-pbact prAttr(2) }

-- Типы контента

id-privAssignRequest OBJECT IDENTIFIER ::= { id-pbactCont privAssignRequest(1) }
id-privAssignResult  OBJECT IDENTIFIER ::= { id-pbactCont privAssignResult(2) }
id-readRequest       OBJECT IDENTIFIER ::= { id-pbactCont readRequest(3) }
id-readResult        OBJECT IDENTIFIER ::= { id-pbactCont readResult(4) }
id-compareRequest    OBJECT IDENTIFIER ::= { id-pbactCont compareRequest(5) }
id-compareResult     OBJECT IDENTIFIER ::= { id-pbactCont compareResult(6) }
id-addRequest        OBJECT IDENTIFIER ::= { id-pbactCont addRequest(7) }
id-addResult         OBJECT IDENTIFIER ::= { id-pbactCont addResult(8) }
id-deleteRequest     OBJECT IDENTIFIER ::= { id-pbactCont deleteRequest(9) }
id-deleteResult      OBJECT IDENTIFIER ::= { id-pbactCont deleteResult(10) }
id-modifyRequest     OBJECT IDENTIFIER ::= { id-pbactCont modifyRequest(11) }
id-modifyResult      OBJECT IDENTIFIER ::= { id-pbactCont modifyResult(12) }
id-renameRequest     OBJECT IDENTIFIER ::= { id-pbactCont renameRequest(13) }
id-renameResult      OBJECT IDENTIFIER ::= { id-pbactCont renameResult(14) }


-- Типы атрибутов, несущие определения привилегий

id-at-accessService OBJECT IDENTIFIER ::= { id-pbactPrivAttr 1 }

END
```

Добавление I

Неформальная спецификация профиля синтаксиса криптографических сообщений

(Данное Добавление не является неотъемлемой частью настоящей Рекомендации)

Реализация, поддерживающая только модуль **CmsTelebiometric**, не соответствует спецификации IETF по CMS и не считается спецификацией реализации. Она приводится здесь только для информации и для проверки согласованности.

```
CmsTelebiometric { joint-iso-itu-t(2) telebiometrics(42) th(3) part0(0)
   modules(0) cmsProfile(1) version1(1) }
DEFINITIONS ::=
BEGIN

-- EXPORTS All

IMPORTS

  -- из Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2

      ATTRIBUTE, Attribute{}, DistinguishedName, objectIdentifierMatch
    FROM InformationFramework {joint-iso-itu-t ds(5) module(1)
informationFramework(1) 8}

  -- из Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8

  ALGORITHM, AlgorithmIdentifier, Certificate, CertificateSerialNumber
    FROM AuthenticationFramework {joint-iso-itu-t ds(5) module(1)
authenticationFramework(7) 8}

  -- из Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6

  integerMatch, octetStringMatch
    FROM SelectedAttributeTypes {joint-iso-itu-t ds(5) module(1)
selectedAttributeTypes(5) 8}  ;

CONTENT-TYPE ::= TYPE-IDENTIFIER

ContentType ::= CONTENT-TYPE.&id

ContentInfo ::= SEQUENCE {
  contentType  CONTENT-TYPE.&id ({TelebSupportedcontentTypes}),
  content      CONTENT-TYPE.&Type
          ({TelebSupportedcontentTypes}{@contentType})OPTIONAL,
  ... }

TelebSupportedcontentTypes CONTENT-TYPE ::=
  { signedData | envelopedData | ct-authEnvelopedData, ...}

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

Attributes { ATTRIBUTE:AttrList } ::=
             SET SIZE (1..MAX) OF Attribute {{ AttrList }}

signedData CONTENT-TYPE ::= {
                SignedData
  IDENTIFIED BY id-signedData }

SignedData ::= SEQUENCE  {
  version               CMSVersion (v3),
```

```
  digestAlgorithms      SET (SIZE (1)) OF AlgorithmIdentifier
                          {{Teleb-Hash-Algorithms}},
  encapContentInfo      EncapsulatedContentInfo,
  certificates     [0]  IMPLICIT SET (SIZE (1..MAX)) OF Certificate OPTIONAL,
--crls             [1]  IMPLICIT RevocationInfoChoices OPTIONAL,
  signerInfos           SignerInfos,
  ... }

Teleb-Hash-Algorithms ALGORITHM ::= {...}

EncapsulatedContentInfo ::= SEQUENCE {
  eContentType     CONTENT-TYPE.&id({IncludedContent}),
  eContent     [0] EXPLICIT OCTET STRING
    (CONTAINING CONTENT-TYPE.&Type({IncludedContent}
     {@eContentType})) OPTIONAL }

IncludedContent CONTENT-TYPE ::= {envelopedData, ...}

SignerInfos ::= SET (SIZE (1)) OF SignerInfo

SignerInfo ::= SEQUENCE {
  version                 CMSVersion (v1),
  sid                     SignerIdentifier,
  digestAlgorithm         AlgorithmIdentifier {{Teleb-Hash-Algorithms}},
  signedAttrs        [0]  IMPLICIT Attributes{{SignedAttributes}} OPTIONAL,
  signatureAlgorithm      AlgorithmIdentifier {{Teleb-Signature-Algorithms}},
  signature               SignatureValue,
  unsignedAttrs      [1]  IMPLICIT Attributes {{UnsignedAttributes}} OPTIONAL,
  ... }

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

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

SignedAttributes ATTRIBUTE ::= { contentType | messageDigest, ... }

Teleb-Signature-Algorithms ALGORITHM ::= {...}

SignatureValue ::= OCTET STRING

UnsignedAttributes ATTRIBUTE ::= {...}

envelopedData CONTENT-TYPE ::= {
                EnvelopedData
  IDENTIFIED BY id-envelopedData }

EnvelopedData ::= SEQUENCE {
  version                   CMSVersion(v0 | v2),
--originatorInfo         [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  encryptedContentInfo      EncryptedContentInfo,
  ...,
  [[2: unprotectedAttrs  [1] IMPLICIT Attributes
       {{UnprotectedAttributes}} OPTIONAL ]] }

RecipientInfos ::= SET SIZE (1) OF RecipientInfo

UnprotectedAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }

RecipientInfo ::= CHOICE {
```

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

KeyAgreeRecipientInfo ::= SEQUENCE {
  version                 CMSVersion (v3),
  originator         [0] EXPLICIT OriginatorIdentifierOrKey,
  ukm                [1] EXPLICIT UserKeyingMaterial OPTIONAL,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  recipientEncryptedKeys RecipientEncryptedKeys,
  ... }

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

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

SupportedDHPublicKeyAlgorithms ALGORITHM ::= {...}

UserKeyingMaterial ::= OCTET STRING (SIZE (64))

KeyEncryptionAlgorithmIdentifier ::=
   AlgorithmIdentifier{{SupportedKeyIncryptAlgorithms}}

SupportedKeyIncryptAlgorithms ALGORITHM ::= {...}

RecipientEncryptedKeys ::= SEQUENCE (SIZE (1)) OF RecipientEncryptedKey

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

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

EncryptedKey ::= OCTET STRING

EncryptedContentInfo ::= SEQUENCE {
  contentType                CONTENT-TYPE.&id ({EncryptedContentSet}),
  contentEncryptionAlgorithm SEQUENCE {
    algorithm                  ALGORITHM.&id ({SymmetricEncryptionAlgorithms}),
    parameter                  ALGORITHM.&Type
            ({SymmetricEncryptionAlgorithms}{@.algorithm})} OPTIONAL,
  encryptedContent       [0] IMPLICIT EncryptedContent OPTIONAL,
  ... }

EncryptedContentSet CONTENT-TYPE ::= {...}

SymmetricEncryptionAlgorithms ALGORITHM ::= {...}

EncryptedContent ::= OCTET STRING

ct-authEnvelopedData CONTENT-TYPE ::= {
               AuthEnvelopedData
```

```
  IDENTIFIED BY id-ct-authEnvelopedData }

AuthEnvelopedData ::= SEQUENCE {
  version                   CMSVersion (v0),
--originatorInfo        [0] IMPLICIT OriginatorInfo OPTIONAL,
  recipientInfos            RecipientInfos,
  authEncryptedContentInfo EncryptedContentInfo,
  authAttrs               [1] IMPLICIT Attributes {{AuthAttributes}} OPTIONAL,
  mac                       MessageAuthenticationCode,
  unauthAttrs             [2] IMPLICIT Attributes {{UnauthAttributes}} OPTIONAL }

AuthAttributes ATTRIBUTE ::= {...}

MessageAuthenticationCode ::= OCTET STRING

UnauthAttributes ATTRIBUTE ::=
  { aa-CEKReference | aa-CEKMaxDecrypts | aa-KEKDerivationAlg }

KEKRecipientInfo ::= SEQUENCE {
  version                 CMSVersion (v4),
  kekid                   KEKIdentifier,
  keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
  encryptedKey            EncryptedKey }

KEKIdentifier ::= SEQUENCE {
  keyIdentifier OCTET STRING,
--дата          GeneralizedTime OPTIONAL,
--прочее        OtherKeyAttribute OPTIONAL,
  ... }

contentType ATTRIBUTE ::= {
  WITH SYNTAX             CONTENT-TYPE.&id({envelopedData, ...})
  EQUALITY MATCHING RULE objectIdentifierMatch
  SINGLE VALUE            TRUE
  ID                      id-contentType }

messageDigest ATTRIBUTE ::= {
  WITH SYNTAX             OCTET STRING
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE            TRUE
  ID                      id-messageDigest }

aa-CEKReference ATTRIBUTE ::= {
  WITH SYNTAX             CEKReference
  EQUALITY MATCHING RULE octetStringMatch
  SINGLE VALUE            TRUE
  ID                      id-aa-CEKReference }

CEKReference ::= OCTET STRING

aa-CEKMaxDecrypts ATTRIBUTE ::= {
  WITH SYNTAX             CEKMaxDecrypts
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE            TRUE
  ID                      id-aa-CEKReference }

CEKMaxDecrypts ::= INTEGER

aa-KEKDerivationAlg ATTRIBUTE ::= {
  WITH SYNTAX             KEKDerivationAlgorithm
  EQUALITY MATCHING RULE integerMatch
  SINGLE VALUE            TRUE
  ID                      id-aa-KEKDerivationAlg }
```

```
KEKDerivationAlgorithm ::= SEQUENCE {
  kekAlg       AlgorithmIdentifier {{SupportedKeyIncryptAlgorithms}},
  pbkdf2Param  PBKDF2-params }

PBKDF2-params ::= SEQUENCE {
  salt CHOICE {
    specified   OCTET STRING,
--  otherSource AlgorithmIdentifier {{PBKDF2-SaltSources}}
    ... },
  iterationCount INTEGER (1..MAX),
  keyLength      INTEGER (1..MAX) OPTIONAL,
  prf            AlgorithmIdentifier {{PBKDF2-PRFs}},
  ... }

PBKDF2-PRFs ALGORITHM ::= {...}

id-pkcs OBJECT IDENTIFIER ::=
   { iso(1) member-body(2) usa(840) rsadsi(113549) pkcs(1) }

id-pkcs-9 OBJECT IDENTIFIER ::= { id-pkcs pkcs-9(9) }

id-ct OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) ct(1) }
id-aa OBJECT IDENTIFIER ::= { id-pkcs-9 smime(16) attributes(2) }

id-contentType         OBJECT IDENTIFIER ::= { id-pkcs-9 3 }
id-messageDigest       OBJECT IDENTIFIER ::= { id-pkcs-9 4 }
id-aa-CEKReference     OBJECT IDENTIFIER ::= { id-aa 30 }
id-aa-CEKMaxDecrypts   OBJECT IDENTIFIER ::= { id-aa 31 }
id-aa-KEKDerivationAlg OBJECT IDENTIFIER ::= { id-aa 32 }

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

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

id-ct-authEnvelopedData OBJECT IDENTIFIER ::= { id-ct 23 }

END -- CmsTelebiometric
```

Библиография

[b-ITU-T X.841] Recommendation ITU-T X.841 (2000) | ISO/IEC 15816:2002, *Information technology – Security techniques – Security information objects for access control*

[b-IEC 62351-8] IEC/TS 62351-8:2011, *Power systems management and associated information exchange – Data and communications security − Part 8: Role based access control*

[b-NIST 800-56A] NIST Special Publication 800-56A, Revision 2 (2013), *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*

[b-NIST 800-162] NIST Special Publication 800-162 (2014), *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*

[b-IETF RFC 5480] IETF RFC 5480 (2009), *Elliptic Curve Cryptography Subject Public Key Information*

[b-IETF RFC 7191] IETF RFC 7191 (2014), *Cryptographic Message Syntax (CMS) - Key Package Receipt and Error Content Types*

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия D	Принципы тарификации и учета, а также экономические и политические вопросы, связанные с международными услугами в области электросвязи/ИКТ
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и среда передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Окружающая среда и ИКТ, изменение климата, электронные отходы, энергоэффективность; конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	Управление электросвязью, включая СУЭ и техническое обслуживание сетей
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация, а также соответствующие измерения и испытания
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	**Сети передачи данных, взаимосвязь открытых систем и безопасность**
Серия Y	Глобальная информационная инфраструктура, аспекты межсетевого протокола, сети последующих поколений, интернет вещей и "умные" города
Серия Z	Языки и общие аспекты программного обеспечения для систем электросвязи

Отпечатано в Швейцарии
Женева, 2017 г.