国 际 电 信 联 盟

ITU-T X.1080.0

(03/2017)

国际电信联盟
电信标准化部门

X系列：数据网、开放系统通信和安全性

网络空间安全 – 反垃圾信息

电子生物特征数据保护的访问控制

ITU-T X.1080.0 建议书

国际电信联盟

ITU-T X系列建议书

数据网、开放系统通信和安全性

公用数据网	X.1–X.199
开放系统互连	X.200–X.299
网间互通	X.300–X.399
报文处理系统	X.400–X.499
号码簿	X.500–X.599
OSI组网和系统概貌	X.600–X.699
OSI管理	X.700–X.799
安全	X.800–X.849
OSI应用	X.850–X.899
开放分布式处理	X.900–X.999
信息和网络安全	
一般安全问题	X.1000–X.1029
网络安全	X.1030–X.1049
安全管理	X.1050–X.1069
生物测定	**X.1080–X.1099**
安全应用和服务	
组播安全	X.1100–X.1109
家庭网络安全	X.1110–X.1119
移动安全	X.1120–X.1139
网页安全	X.1140–X.1149
安全协议	X.1150–X.1159
对等网络安全	X.1160–X.1169
网络身份安全	X.1170–X.1179
PITV安全	X.1180–X.1199
网络空间安全	
计算网络安全	X.1200–X.1229
反垃圾信息	X.1230–X.1249
身份管理	X.1250–X.1279
安全应用和服务	
应急通信	X.1300–X.1309
泛在传感器网络安全	X.1310–X.1339
PKI相关建议书	X.1340–X.1349
物联网（IoT）安全	X.1360–X.1369
智能交通系统（ITS）安全	X.1370–X.1379
网络安全信息交换	
网络安全综述	X.1500–X.1519
脆弱性/状态信息交换	X.1520–X.1539
事件/事故/探索法信息交换	X.1540–X.1549
政策的交换	X.1550–X.1559
探索法和信息要求	X.1560–X.1569
标示和发现	X.1570–X.1579
确保交换	X.1580–X.1589
云计算安全	
云计算安全综述	X.1600–X.1601
云计算安全设计	X.1602–X.1639
云计算安全最佳实践和指导原则	X.1640–X.1659
云计算安全实现	X.1660–X.1679
其他云计算安全	X.1680–X.1699

欲了解更详细信息，请查阅ITU-T建议书目录。

ITU-T X.1080.0 建议书

电子生物特征数据保护的访问控制

摘要

ITU-T X.1080.0建议书提供了如何保护电子生物特征信息免受未经授权访问的规范。它通过面向服务的观点来实现，仅提供某个特定目的所需的必要信息，例如，访问不仅建立在有权知道的基础上，而且建立在按需知道的基础上。

本建议书的核心是一个属性规范，它包含在一个属性证书或公开密钥证书中，用于详细规定针对一种或多种服务类型某个特定的实体有哪些权限。

安全性通过使用一个加密消息语法（CMS）概要文件来提供。CMS概要文件规定了认证、完整性，以及需要的话还有机密性（加密）。

该概要文件旨在为电子生物特征通用规范提供安全支持。该概要文件假定并取决于公开密钥基础设施（PKI）的正确部署。

ITU-T X.1080.0建议书也取决于权限管理基础设施（PMI）的部署。

沿革

版本	建议书	批准日期	研究组	唯一ID*
1.0	ITU-T X.1080.0	2017-03-30	17	11.1002/1000/13193

关键词

访问控制、Diffie-Hellman算法、公开密钥基础设施（PKI）、电子生物特征

* 为获取本建议书，请在网页浏览器内键入URLhttp://handle.itu.int/，然后输入唯一ID。例如： http://handle.itu.int/11.1002/1000/11830-en。

前言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定ITU-T各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA第1号决议规定了批准建议书须遵循的程序。

属ITU-T研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简明扼要起见而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联尚未收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能并非最新信息，因此特大力提倡他们通过下列网址查询电信标准化局（TSB）的专利数据库：http://www.itu.int/ITU-T/ipr/。

目录

引言

在收集个人电子生物特征数据时，有可能侵犯隐私。

因若干原因而要做好此类信息的保护工作。有了信息，可访问某个公司或组织，或者因其敏感性而需限制其散布。

防止不必要的泄露电子生物特征数据涉及两个主要方面：

– 在传输过程中做好数据的保护工作，这通常可以通过加密手段来实现，并保护好所存储的数据；以及

– 控制对所存储数据的访问。

就机密性（加密）、认证、完整性、物理保护、防火墙使用、病毒保护程序等而言，电子生物特征系统应具备高水平的安全性，同时，对所存储的数据的访问而言，还需要建立一套精密的访问控制系统，尤其对有关个人的特殊信息。对电子生物特征系统而言，后者显得尤为重要。

通用的访问控制方案存在一个缺陷，它们主要考虑的是关于信息的*有权知道*（或拒绝）问题，但并不十分关心关于*按需知道*的问题。*按需知道*指的是有权看到数据并不足够，，还必须明确，该信息需用于合法的目的。

信息应只供其预期目的使用。收集病人的医疗信息，为的是为该病人提供最佳的治疗，而不应将这些信息用于任何其他的目的，可能的例外情况是用在严格控制下的研究项目中，这些项目可能需要用到部分收集自特定病患人群的信息。否则，应做好信息的保护工作，以防信息被搜罗。

有两种主要类型的访问控制：物理的和逻辑的。物理访问控制限制进入校园、建筑物、房间和物理的信息技术（IT）资产。逻辑访问限制连入计算机网络、系统文件和数据。本建议书只讨论有关逻辑访问控制问题。

访问控制包括通过服务提供者提供的、关于访问者安全认证的服务。本建议书假定使用数字签名和一个已建立的公开密钥基础设施（PKI）。

ITU-T X.1080.0建议书可供其他电子生物特征规范参考。

附件A，构成本建议书的一个组成部分，说明了ITU-T X.1080系列建议书所用之对象标识符的分配。

附件B，构成本建议书的一个组成部分，提供了本建议书使用的、有关IETF RFC 5652所述的加密消息语法（CMS）的电子生物特征概要文件。它也可供其他电子生物特征规范参考。

附件C，构成本建议书的一个组成部分，以抽象句法表示法1（ASN.1）模块的形式，提供了关于权限声明和分配协议的正式规范。

附录I，不构成本建议书的一个组成部分，以ASN.1模块的形式，提供了关于加密消息语法（CMS）概要文件的非正式规范。

ITU-T X.1080.0 建议书

电子生物特征数据保护的访问控制

1 范围

本建议书提供了如何使用基于隐私的电子生物特征访问控制（ACT），在电子生物特征环境中做好隐私保护的规范。虽然本建议书不能确定所有可能的信息类型，但它在提供通用工具以安全的方式处置所有类型的信息的范畴内。这包括使用权限声明定义关于分配权限的协议以及关于访问信息的协议。本建议书提供了导则，但并不包含合规要求。

下列问题不在本建议书的讨论范围之内：

– 信息的物理保护；

– 负责维护安全系统的操作人员未经授权的访问，这种未经授权的访问有可能绕过安全系统而造成安全系统失去作用。

2 参考文献

下列ITU-T建议书和其他参考文献的条款，通过在本文本中的引用而构成本建议书的条款。在出版时，所指出的版本是有效的。所有的建议书和其他参考文献均面临修订；因此鼓励本建议书的使用者探讨使用下列建议书和其他参考文献最新版本的可能性。当前有效的ITUT-T建议书清单将定期出版。

在本建议书中引用某个文件时，并未给予该文件作为一份独立建议书的地位。

[ITU-T X.500系列] ITU-T X.5xx建议书 (2016) | ISO/IEC 9594-x系列，信息技术 — 开放系统互连 – 号码簿。

[ITU-T X.501] ITU-T X.501建议书 (2016) | ISO/IEC 9594-2，信息技术 — 开放系统互连 – 号码簿：模型。

[ITU-T X.509] ITU-T X.509建议书(2016) | ISO/IEC 9594-8， 信息技术 — 开放系统互连 – 号码簿：公开密钥和属性证书框架。

[ITU-T X.520] ITU-T X.520建议书(2016) | ISO/IEC 9594-6， 信息技术 — 开放系统互连 – 号码簿：选择的属性类型。

[ITU-T X.521] ITU-T X.521建议书(2016) | ISO/IEC 9594-7， 信息技术 — 开放系统互连 – 号码簿: 选择的对象类别。

[ITU-T X.680] ITU-T X.680建议书(2015) | ISO/IEC 8824-1， 信息技术— 抽象语法记法一（ASN.1）：基本记法规范。

[ITU-T X.681] ITU-T X.681建议书(2015) | ISO/IEC 8824-2， 信息技术— 抽象语法记法一（ASN.1）：信息对象规范。

[ITU-T X.682] ITU-T X.682建议书(2015) | ISO/IEC 8824-3，信息技术— 抽象语法记法一（ASN.1）：约束规范。

[ITU-T X.683] ITU-T X.683建议书(2015) | ISO/IEC 8824-4，信息技术— 抽象语法记法一（ASN.1）：ASN.1规范的参数化。

[ITU-T X.690] ITU-T X.690建议书(2015) | ISO/IEC 8825-1，信息技术 — ASN.1编码规则：基本编码规则（BER）、规范编码规则（CER）和唯一编码规则（DER）。

[ITU-T X.1080.1] ITU-T X.1080.1建议书 (2011)，电子医疗和全球远程医疗 - 通用通信协议。

[ITU-T X.1081] ITU-T X.1081建议书 (2011)，电子生物特征多模模型 – 关于电子生物特征安全方面的规范框架。

[IETF RFC 2631] IETF RFC 2631 (1999)，Diffie-Hellman密钥协议方法。

[IETF RFC 3185] IETF RFC 3185 (2001)，重用CMS内容加密密钥。

[IETF RFC 5083] IETF RFC 5083 (2007)，加密消息语法（CMS） — 经过认证的 — 经过封装的-数据内容类型。

[IETF RFC 5652] IETF RFC 5652 (2009)，加密消息语法（CMS）。

[IETF RFC 5753] IETF RFC 5753 (2010)，在加密消息语法（CMS）中使用椭圆曲线加密（ECC）算法。

[IETF RFC 5911] IETF RFC 5911 (2010)，关于加密消息语法（CMS）和S/MIME的ASN.1新模块。

[IETF RFC 6268] IETF RFC 6268 (2011)，关于加密消息语法（CMS）和使用X.509（PKIX）的公开密钥基础设施的额外的ASN.1新模块。

3 定义

3.1 他处定义的术语

本建议书采用了下列他处定义的术语：

3.1.1 属性证书（attribute certificate） [ITU-T X.509]：一种数据结构，由属性机构数字签署，它将某些属性值与其持有者的身份信息进行绑定。

3.1.2 属性类型（attribute type） [ITU-T X.501]：是属性的一个构件，指明该属性所持有的信息类别。

3.1.3 属性值（attribute value） [ITU-T X.501]：由一个属性类型所指明的信息类的一个具体实例。

3.1.4 权限（privilege） [ITU-T X.509]：由一个机构分配给一个实体的一个属性或特性。

3.1.5 权限持有者（privilege holder） [ITU-T X.509]：一个已被赋予权限的实体。权限持有者可以出于某个特殊目的来声明其权限。

3.1.6 权限验证者（privilege verifier） [ITU-T X.509]：一个依据某项权限策略对证书进行验证的实体。

3.1.7 机构源（source of authority）（SOA） [ITU-T X.509]：一个属性机构，某个特殊资源的权限验证者将之看作是分配一系列权限以生命该资源的最终机构。

3.2 本建议书定义的术语

本建议书定义了下列术语：

3.2.1 访问控制（access control）：一种安全技术，用来规定什么人可以对计算环境中的信息资源做什么。

3.2.2 访问服务（access service）：由服务提供者提供的一种服务，用来执行某项特定的业务。

3.2.3 访问者（accessor）：一个权限持有者，它使用其权限来访问某项特定的访问服务。

3.2.4 属性（attribute）：与某个对象关联的、某种特定类型信息的一个片段。由各属性组成与某个对象关联的信息。

3.2.5 数据保护域（data protection domain）：一个域，要保护的信息在单一管理组件中。

3.2.6 唯一名称（distinguished name）：用于确定某个对象的名称，在某个特定的上下文中，它是独一无二的，它由一个或多个名称的构件构成，反映该对象在由各对象组成的层次结构中所处的位置。

3.2.7 对象（object）：一个人、一个部门、一个专业或其他类型的对象，存在关于它的信息，并可通过唯一的名称来识别它。

3.2.8 对象类（object class）：共享某些特性的、确定的实体家族。

3.2.9 操作（operation）：一种相互作用，出于某个特定的目的，由访问者与服务提供者之间的请求和应答组成。

3.2.10 规范（specification）：一个ITU-T建议书、一个国际标准或者由一个公认的标准制定组织（SDO）制定的任何规范。

4 缩写词和首字母缩略语

本建议书采用下列缩写词和首字母缩略语：

AA 属性机构
ABAC 基于属性的访问控制
ACL 访问控制列表
ACT 电子生物特征的访问控制
AES 高级加密标准
ASN.1 抽象语法记法一
BER 基本编码规则
CA 认证机构
CEK 内容加密密钥
CMS 加密消息语法
DER 唯一编码规则
DH Diffie-Hellman算法
ECC 椭圆曲线密码术

ECDH 椭圆曲线Diffie-Hellman算法

GCM Galois/计数器模式

KEK 密钥加密密钥

LDAP 轻量级目录访问协议

MAC 消息认证码

PDU 协议数据单元

PKI 公开密钥基础设施

PMI 权限管理基础设施

SDO 标准制定组织

SOA 机构源

5 排印惯例

本建议书使用粗体的courier新字体来表示抽象句法标记法1（ASN.1）记法。若在常规文本中要表示ASN.1 的类型和值时，为了区别于常规文本，将使用粗体的courier新字体来表示它们。

如果列表中的各项以数字标识（而不是以“–”或字母标识），那么这些项应被认为是一个过程中的各个步骤。

6 基本概念和模型

6.1 单个数据保护域中的保护

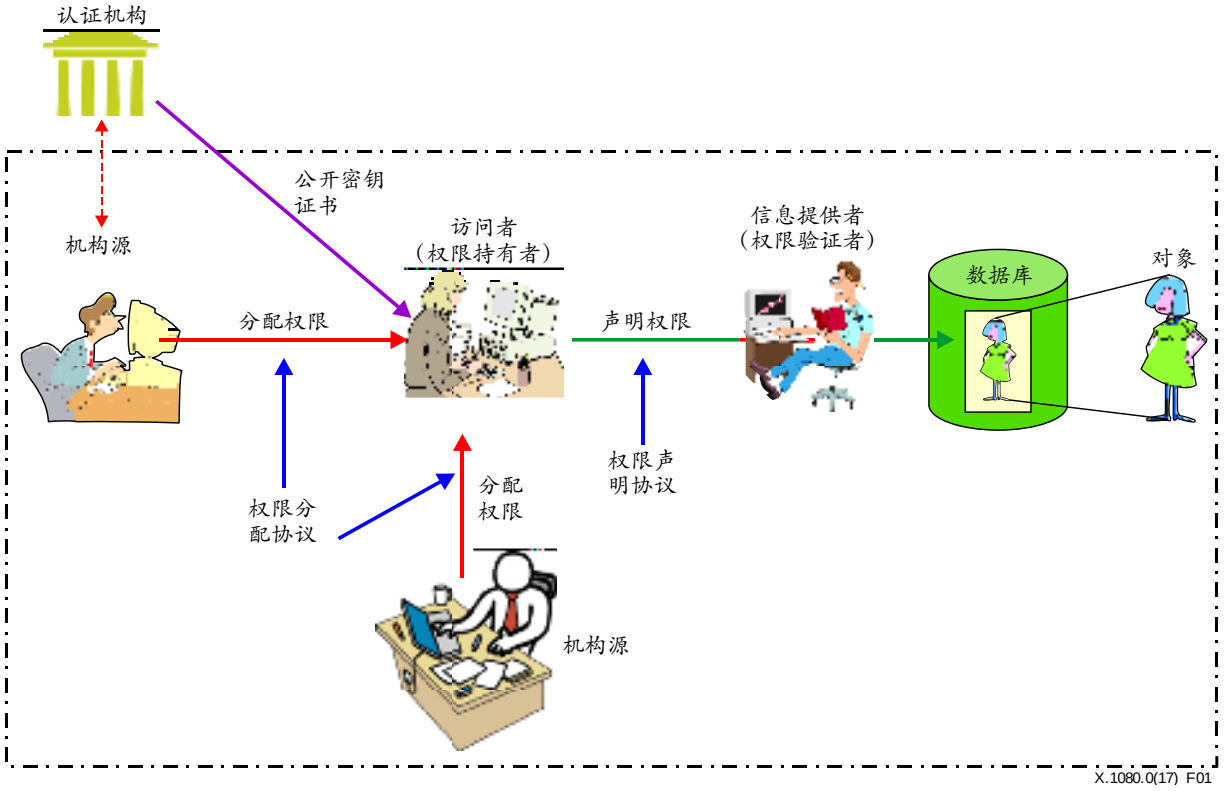

图 1 – 单个数据保护域模型

图1说明了在单个数据保护域内某个访问控制环境中各种合作伙伴和协议。一个数据保护域包含保护中涉及的所有实体，它们接受共同的管理。

称为访问者的实体可有一项任务功能，它需要许可，以便对信息提供者持有的关于对象的信息执行操作。例如，负责某病人的医生需要访问该病人的健康记录，以便给予最佳的治疗，而其他与此病人无关的医生就不需要拥有这些信息。

对信息执行某个特定操作的许可，应只提供给拥有权限并真正需要执行该操作的访问者，即它已被分配必要的权限。

规定在什么条件下向实体分配权限已超出了本建议书的讨论范围。讨论范围仅限提供必要的工具以便以安全的方式来管理权限。

一个数据保护域需要建立一个或多个机构来向实体分配权限。在[ITU-T X.509]中定义的机构源（SOA）一词用在此处，原因是有关为某个特定领域分配权限的最终机构一词在数据保护域内。

本建议书使用证书来向访问者分配权限。权限可承载于属性构件的属性证书中或者subjectDirectoryAttributes扩展的公开密钥证书中。长久的权限通常可在公开密钥证书中提供，而临时权限通常可在属性证书中提供。

图1说明了单个数据保护域内各种构件之间的关系。它显示了两个SOA，每个负责为权限持有者的不同方面问题分配权限。

访问者的日常操作可能需要某些权限，因此而具更长久的特性；其他的权限则分配用于处置特殊的情况，因此而具临时的特性。

当权限分配给某个访问者时，访问者变成为权限的持有者，并可利用其权限，通过权限声明协议来访问信息。在允许对信息进行访问之前，代表信息提供者的权限验证者对所声明的权限进行检查。

一个SOA可通过本地方式或者通过将之嵌入权限分配协议的签名属性证书中，来提供权限，如图1所示。

6.2 交叉数据保护域

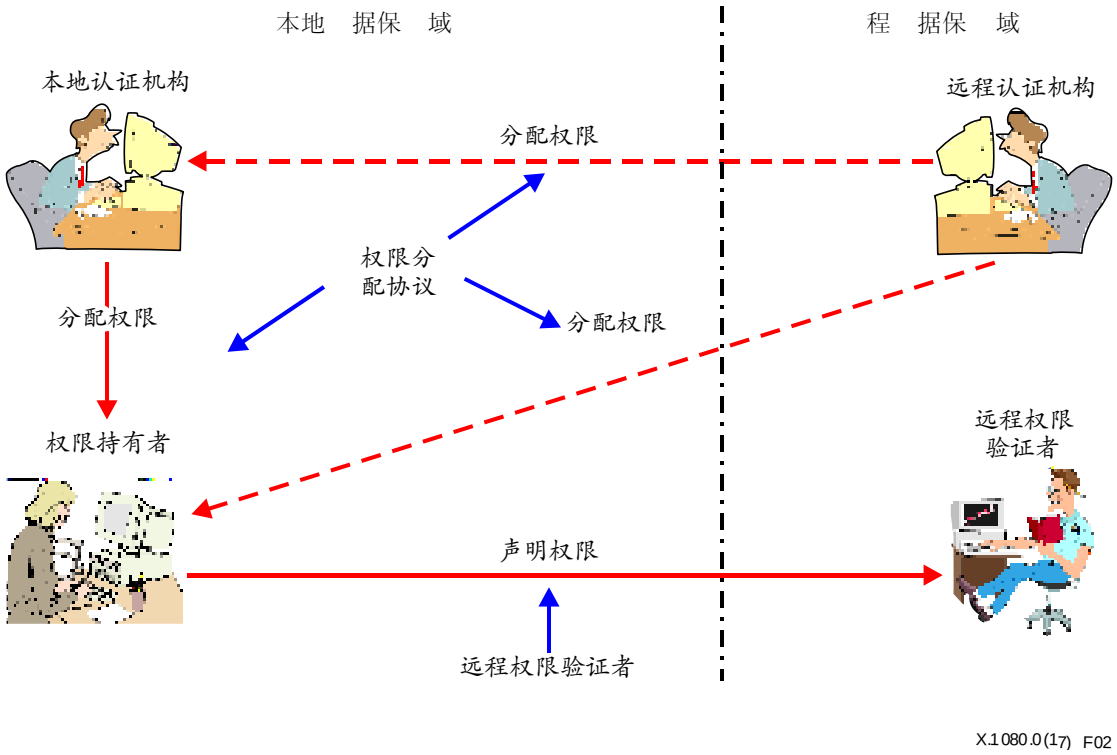

图 2 – 交叉数据保护域模型

图2说明了访问者需要访问另一个数据保护域中信息的情况，例如，在治疗一个病人时，其关键的医疗信息位于另一个数据保护域中。

在这种环境下，访问者直接或者通过一个本地SOA来请求访问关于某个特定对象的特定信息。该请求如何传送超出了本建议书的讨论范围，但它可以通过电子邮件或者通过电话来传送。

远程SOA以持有必要权限的accessService属性类型来生成一个属性证书。该属性证书通过远程SOA来签署。属性证书的持有者可以是本地SOA或者需要权限的实体。

如果是拥有所分配权限的本地SOA，那么该SOA：

a) 如果需要的话，对内容进行解密。

b) 对有关内容的签名进行验证。

c) 从内容中提取属性证书。

d) 对提取的属性证书进行验证。

e) 利用取自收到的属性证书中的权限信息，以及通过需要权限作为持有者的实体，创建一个新的属性证书。

f) 然后将该属性证书连同收自远程SOA的属性证书传送给需要权限的实体（如权限持有者）。

6.3 面向服务的模型

就一个访问者可调用一组不同的函数（称为*访问服务*）而言，电子生物特征的访问控制是面向服务的。一个访问服务指的是由能执行特定访问服务的服务提供者提供的一种功能。访问服务的一个例子是操控医院内患者信息的能力。一个对象标识符用于确定一个特定的访

问服务类型。本建议书不定义具体的访问服务，但为建立访问服务提供工具。引用规范可定义与其范围相关的访问服务。

尽管一个访问者可访问一个特定的服务，但他们可能没有权限涵盖该服务的所有方面。

6.4 对象和属性模型

[ITU-T X.501]中定义的信息模型是为了应对不同的数据结构。本建议书将该信息模型用于访问控制。

在ITU-T X.501模型中，待保护对象在对象类中进行组织，当中，一个对象类中的各对象拥有共同的特征，在某个特定的上下文中它们是相关的。一个对象类通过一个对象标识符来确定。通过将对象标识符分配给一组与电子生物特征相关的类别的对象类，[ITU-T 1080.1]扩展了该概念。

[ITU-T X.521]还定义了一组通用的、可用的对象类。当需要一个不直接与电子生物特征相关的对象类时，应尽可能使用一个已经定义的对象类。

对单个对象，将通过[ITU-T X.501]定义的唯一的名称来确定。唯一的名称由一个或多个名称构件组成，反映其在一个对象层次结构中的位置。

与一个对象相关的信息被建模为一组属性。带有共同特征的属性构成一个属性类型。一个属性类型通过一个对象标识符来确定。一个属性是对象标识符（用于确定属性类型以及该类型的一个或多个值）的一个序列。一个对象可能只有一个某种特定类型的属性。[ITU-T X.520]定义了一组通用的、可用的属性类型。应尽可能使用已经定义的属性类型。根据需要，额外的属性类型可通过单个规范来定义。

本建议书的单个用法将需规定该信息模型与实际数据基础结构之间的某种映射。如果信息保持在一个轻量级的目录访问协议（LDAP）目录或者一个安全的、基于[ITU-T X.500系列]中规范的目录中，那么这种映射将变得便利起来。

6.5 访问控制基本原则

本节的目的是对建立电子生物特征访问控制的一般原则做一概述。传统访问控制主要关注的是基于某些固定的参数的有权访问数据或拒绝访问数据问题。本建议书采用一种经过扩展的方法，也考虑到了“按需知道”方面的问题。这是通过在第6.3节中所述的服务方法来实现的。为执行某个特定的操作，一个访问者需要拥有允许访问相关访问服务的权限以及允许访问所需信息的权限。

该权限包括对一个或多个对象类的对象的访问，可能仅限于一些命名对象。可以执行的操作类型对不同的对象类和命名对象可能是不同的。

针对对象的操作可以包括针对单个属性和属性值的操作。允许的操作对不同的属性类型可能是不同的。

6.6 与其他访问控制方案的关系

有各种类型的访问控制，来覆盖不同的访问控制需求。此处的意图是对不同的访问控制方案做简短描述，并将之与ACT相关联。

6.6.1 ITU-T X.501定义的基本访问控制

[ITU-T X.501]定义的基本访问控制用于保护目录中的信息，如[ITU-T X.500系列]所定义的那样。它提供了详细的访问控制列表，详细规定了如何访问以及允许什么样的不同用户访问。有别于本建议书，它只规定了有关“有权知道”的问题，并未涉及有关“按需知道”的问题。

6.6.2 基于规则的访问控制

[ITU-T X.501]和[b-ITU-T X.841]都定义了一种基于规则的访问控制类型。在该类型的访问控制中，对要保护的数据，用需要什么样的保护这样的信息来做标记，同时访问用户已对访问请求中相关的信息进行了认证，规定了与访问特定信息有关的许可证等级。该概念与本建议书的不同之处在于需要标记所存储的数据项，且只规定了有关“有权知道”的问题，并未涉及有关“按需知道”的问题。

6.6.3 为智能网格定义的、基于角色的访问控制

基于角色的访问控制，如[b-IEC 62351-8]中规定的那样，主要关注的是用户以及用户执行的任务。一个角色指的是一组有关对象的权利（可以对特定目标执行的行动）。一个用户可拥有一个或几个角色。对访问控制而言，角色是一种中介，通过减小访问控制信息的粒度，来减少访问控制列表（ACL）所需的信息量。未为每个用户分别列出针对系统对象的许可，但为用户提供了角色，对每个角色的权利只做一次描述。

6.6.4 基于属性的访问控制

基于属性的访问控制（ABAC）是一个逻辑访问控制模型，它是可辨识的，原因是，它通过依据实体（主体和客体）的属性、操作和请求相关的环境对规则进行评估，来控制对对象的访问。ABAC系统能够执行自主访问控制和强制访问控制概念。ABAC允许精确的访问控制，使得更多数量的离散输入能进入一个访问控制决策，提供一组更大的、有关这些变量的可能组合，从而反映一组更大的和更明确的可能规则，以实现对策略的表述。

关于ABAC的一个很好的介绍，请参见[b-NIST 800-162]。

6.7 协议概述

6.7.1 权限评估协议

权限评估协议包含一组加密消息句法（CMS）内容类型。每种访问类型需要一个请求内容类型和一个结果内容类型。

使用CMS来传送内容类型的实例，如附件B中所述。在附件C中提供了一个正式的ASN.1模块。

6.7.2 权限分配协议

权限分配协议用于在SOA与SOA之间或者从SOA到权限持有者的属性证书传送。

为本协议定义了一个内容类型对，一个内容类型用于以属性证书形式传送权限，一个内容类型用于报告结果。

使用CMS来传送内容类型的实例，如附件B中所述。在附件C中提供了一个正式的ASN.1模块。

6.8 CMS的使用

在附件B中提供了一个关于电子生物特征使用CMS的概要文件。

为确保信息来源得以正确验证，通过本建议书定义的类型内容将被封装在一个**signed Data**内容类型的实例中。由于可能传送敏感信息，因此也建议将之封装在一个**enveloped Data**内容类型的实例中。本建议书不使用**ct-autEnvelopedData**内容类型。

6.9 公开密钥证书注意事项

将由发放者利用其私有密钥签发一个属性证书，并利用对应的、发放给属性证书发放者的公开密钥证书对之进行验证。

原则上，利用signedData内容类型，可将相同的私有密钥用于签署CMS，这将简化验证过程。不过，对不同的目的使用相同的私有密钥，存在安全方面的顾虑。应考虑为两个目的使用不同的私有密钥，但本建议书不对此做强制要求。

7 权限信息的规定

7.1 属性证书的使用

[ITU-T X.509]既允许公开密钥证书也允许属性证书来承载权限信息。在这两种情况下，权限规范均承载于属性中，如[ITU-T X.501]所定义的那样。出于此目的的属性类型有别于用于构建对象信息模型的属性类型。应确保定义用于承载对象信息的属性类型尽可能地简单，用于承载权限信息的属性类型因其性质会相当得复杂。

当使用一个属性证书来承载权限信息时：

a) holder构件确定待分配权限的主体。当一个权限持有者将持有权限的该属性证书提供给一个权限验证者时，权限验证者将对访问者进行验证，以确认访问者确实就是属性证书的权限持有者；

b) issuer构件将持有机构源（SOA）的名称或者某个已被委托来分配权限的属性机构的名称。权限验证者还将获取并验证发放者的公开密钥证书，以验证属性证书的签名；

c) attributes构件将持有accessService类型的一个属性，如第7.3节所定义的那样。

属性证书应由SOA签署，批准已委托发放的权限或AA。

如果持有者和发放者拥有同一认证机构（CA）发放的公开密钥证书，那么这将简化验证过程。

7.2 公开密钥证书的使用

当使用一个公开密钥证书来承载权限信息时，则：

a) subject构件确定已分配权限的访问者。当一个访问者将持有权限的该公开密钥证书提供给一个权限验证者时，权限验证者将对访问者进行验证；

b) issuer构件将持有CA的唯一名称，它负责发放公开密钥证书；

c) subjectDirectoryAttributes扩展将持有accessService属性类型的一个实例（参见第7.3节）。

7.3 访问服务属性类型

7.3.1 访问服务属性语法

计划将accessService类型的一个属性纳入某个属性证书的attributes构件中，或者纳入公开密钥证书的subjectDirectoryAttributes扩展中。它有下列语法：

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

AccessService类型的一个属性提供权限信息，以便某个权限验证者来验证是否应响应访问请求。这是一个多值的属性类型，允许将多个访问服务类型和相关的许可纳入相同的属性中。

一个实体不能使用未包括在该属性中的服务。

accessService属性类型有下列语法：

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

AccessService数据类型的一个值的各构件是：

a) serviceId构件将确定访问者对之有权限的访问服务类型；

b) objectDef构件将规定对象类，对之，权限将分配给有关特定服务类型的属性证书的持有者。它将对每个对象类拥有一个元素，对之，已分配权限。对该访问服务，权限持有者对未由objectDef构件列出的对象类没有任何访问。

7.3.2 对象的选择

对访问者拥有权限的对象的选择，通过ObjectSel数据类型的一个实例来规定。

```
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,
      ... },
    ... },
  ... }
```

ObjectSel数据类型的一个值将规定每个对象类的权限，对之，已为被访问的服务类型分配权限。它有下列构件：

– objectClass构件将规定要分配权限的对象类；

– objSelect构件将规定对象类的对象，对之，已分配权限。它有两种选择方案：

  a) 如果分配的权限平等地适用于类的所有对象，那么将采用allObj选择方案。它将持有TargetSelect数据类型的一个实例；

b) 如果权限仅适用于确定对象类的选定对象，那么将采用objectNames选择方案。它可包含多个元素，每个元素有下列构件：

i) object构件将规定一个或多个已分配权限的对象。它有下列选择方案：

- names选择方案将规定权限使用的一个或多个对象的名称；
- 将对以下对象组采用subtree选择方案，即对象组中的每个对象都有一个等同于该选择方案持有之唯一名称的唯一名称，或者对象组中的每个对象都有等同于该唯一名称的初始名称构件；

ii) 选择构件将持有TargetSelect数据类型的一个实例。

TargetSelect数据类型有下列语法：

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

TargetSelect数据类型有下列两个可选的构件，当中至少一个构件应出现：

a) objOper构件，当出现时，将规定允许的、有关对象类对象或选定对象的操作。如果它不出现，那么不允许对总的对象进行任何操作；

b) attrSel构件，当出现时，将持有AttributeSel数据类型的一个值（参见第7.3.3节）。如果该构件不出现，那么不允许对对象属性进行任何操作。

7.3.3 属性类型的选择

```
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,
      ... },
    ... },
  ... }
```

AttributeSel数据类型规定权限适用的属性类型，它有下列构件：

– attSelect构件有两种选择方案：

a) 如果权限适用于对象的所有属性，那么将选用allAttr选择方案，它有下列构件：

i) attrOper1构件规定可对属性执行的操作；

b) 如果权限仅适用于对象的某些属性，那么将选用attributes选择方案。访问者对未列出的属性类型没有任何权限，且它不知晓此类未列出的属性类型。该选择方案有下列构件：

i) select构件将规定一个或多个权限适用的属性类型；

ii) attrOper2构件规定可对属性执行的操作。

7.4 总的针对对象的操作

下列数据类型用于规定对一个对象允许的操作：

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

将设置read许可，以便允许访问者从一个对象读取信息。

将设置add许可，以便允许访问者添加特定对象类的新对象。对某个特定类的所有对象都需要add许可。对每个将添加至对象的属性，将为属性类型授予add许可（参见第7.5节）。

将设置modify许可，以便访问者允许修改某个现有的对象。访问者将对总的对象节类或者待修改的已命名对象拥有modify许可。如果访问者添加属性，那么它将对所议的属性类型拥有add许可。如果访问者删除属性，那么它将对所议的属性类型拥有delete权限。如果访问者修改属性，那么它将对所议的属性类型拥有modify许可。如果访问者删除属性，那么它将对所议的属性类型拥有deleteValue许可。如果访问者替换属性，那么它将对所议的属性类型拥有replaceAttribute许可。

将设置delete许可，以便允许访问者删除某个现有的对象。访问者将对总的对象类或者待删除的已命名对象拥有delete许可。

将设置rename许可，以便允许访问者重命名现有的对象。访问者将对总的对象类或者待重命名的已命名对象拥有rename许可。

将设置discloseOnError，以便当操作失败时，允许访问者知晓对象是否存在。

7.5 对属性的操作

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

将对每个想要的属性类型设置read许可，以便允许访问者读取此类属性。访问者将对总的相关的对象类或者相关的已命名对象拥有read许可。此外，访问者将对这些对象类的所有属性拥有read许可，或者它将访问相关的属性类型。

将设置compare许可，以便允许访问者对一个或多个属性进行比较。访问者将对总的对象类或者已命名的对象拥有read许可。此外，访问者将对许可之对象类的所有属性拥有compare许可，或者它将对相关的属性类型进行比较。

将设置add许可，以便允许访问者添加一个或多个属性。访问者将对总的对象类或者已命名的对象拥有modify许可。此外，它将对相关的属性类型拥有add许可。

将设置modify许可，以便允许访问者修改特定类型的某个属性。此外，访问者将对总的对象类或者已命名的对象拥有modify许可。

将设置delete许可，以便允许访问者删除一个或多个属性。此外，访问者将对总的对象类或者已命名的对象拥有modify许可。

将设置deleteValue许可，以便允许访问者从属性类型的某个属性中删去一个或多个属性值。此外，访问者将对总的对象类或者已命名的对象拥有modify许可。

将设置replaceAttribute许可，以便允许访问者将某个给定类型的一个属性替换为相同类型的另一个属性。此外，访问者将对总的对象类或者已命名的对象拥有modify许可。

当操作失败时，将为允许知晓某个属性是否存在的访问者设置discloseOnError。此外，它将整体上为对象设置discloseOnError许可。

7.6 错误处置

错误可能因使用CMS而产生，如附件A.5所讨论的那样。当检测到一个错误时，无需做进一步检查。在所请求访问的结果中返回一个CMS错误。

错误也可能因实际的访问请求的结果而产生。

错误通过第8.10节中定义之数据类型AccessdErr的一个实例来报告。

8 权限声明协议

8.1 概述

下列信息对象集包括所有已定义的内容类型，它们通过本建议书中定义的信息对象来表述。

ActContentTypes CONTENT-TYPE ::= {

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

ActContentTypes集规定的内容类型构成了权限声明协议，它包括诸多不同的访问操作，如第8.4节至第8.9节所规定的那样。

8.2 通用请求构件

下列构件包括在所有的请求中：

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

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

常见的请求参数是：

a) attrCert构件，当出现时，将为访问者规定属性证书或者持有权限的属性证书路径。如果该构件不出现，那么将在终端实体公开密钥证书中为访问者提供权限；

b) serviceId构件将规定要调用的服务类型；

c) invokId构件将为调用的首个操作取值0，然后为每个后续的调用操作递增1。它应该有一个范围，以便确保相同的值不会在两个实体之间的一大段通信时间内被重用。规定要对丢失的请求进行检测，并对应答攻击进行检测。在答复中，某个请求的接收者将使用相同的值，以便访问者对结果和相应的请求进行配对。

8.3 访问某项服务

在第8.4节至第8.9节中规定的所有操作类型都需要访问一个特定的服务。权限验证者（接收者）将检查在相关的属性证书或公开密钥证书内分配给访问者的权限是否允许访问所请求的服务。

如果访问者没有调用所请求服务类型的许可或者服务提供者不支持所请求的服务类型，那么将返回一个noSuchService错误代码。

如果访问者有调用服务的许可，那么将检查所请求的操作是否符合服务类型，如果不符合，那么将返回一个invalidOperationForService错误代码。

8.4 读操作

一个读操作包括一个读请求和一个对应的读结果。

一个读请求作为readRequest内容类型的一个实例来承载，读结果作为readResult内容类型的一个实例来承载。

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

访问者使用readRequest内容类型来读取关于某个特定对象的信息。

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

ReadRequest数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有对象（请求关于它的信息）的唯一名称。

b) selection构件将规定访问者请求之信息的类型（参见第8.11节）。

如果请求指定一个未知的对象，那么读请求将失败，并将返回一个noSuchObject错误代码。

如果依据分配给访问者的权限，访问者没有针对对象的read许可，那么读请求将失败。如果未授予read许可，那么若访问者有针对对象的discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchObject错误代码。

对要返回的每个属性，都需要针对属性类型的read许可。如果访问者没有针对某个特定属性类型的read许可，那么不在结果中返回有关该类型的属性。如果结果是没有任何属性返回，那么请求失败。如果访问者针对所有的属性请求都有discloseOnError许可，那么将返回一个insufficientAccessRight错误代码。否则，将返回一个noInformation错误代码。

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

The privilege verifier shall use an instance of the **readResult** content type to return either the requested information or to report an error situation.

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

ReadResult数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有对象（请求关于它的信息）的名称；

b) 结果构件将持有读请求的结果。它有两种选择方案：

– 如果要返回信息，那么将选择success选择方案，并将持有ObjectInformation数据类型的一个实例（参见第8.12节）。返回的信息是访问者请求的信息与允许检索的信息之间的交集。

– 如果报告一个错误，那么将选择failure选择方案。

8.5 比较操作

一个比较操作包括一个比较请求和一个对应的比较结果。

一个比较请求作为compareRequest内容类型的一个实例来承载，比较结果作为compareResult内容类型的一个实例来承载。

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

使用compareRequest内容类型的一个实例来对属于某种特定属性类型的、出现的一个声称值与属于某个特定对象的、相同类型的一个属性值进行比较。

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

CompareRequest数据类型规定实际的内容，包括下列超出第8.2节定义之范围的构件：

a) object构件将持有对象（将对它的一个属性值进行比较）的唯一名称；

b) Purported构件将持有一个属性类型和属性值的组合，将与所议对象持有的相同类型的一个属性进行比较。

如果请求规定一个未知的对象，那么比较请求将失败，并将返回一个noSuchObject错误代码。

如果依据分配给访问者的权限，访问者没有针对对象的read许可，那么比较请求将失败。如果未授予对象read许可，那么若访问者有针对对象的discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchObject错误代码。

如果访问者没有针对所议属性类型的compare许可，那么比较请求将失败。如果访问者有针对属性类型的discloseOnError许可，那么将返回一个insufficientAccessRight错误代码。否则，将返回一个noInformation错误代码。

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

权限验证者将使用compareResult内容类型的一个实例来返回所请求的信息或报告错误情况。

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

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

CompareResult数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有对象的唯一名称，对该对象提出比较请求；

b) 结果构件将持有访问请求的结果。它有两种选择方案：

– 如果选择success选择方案，那么将以下列构件返回CompareOK数据类型的一个实例：

i) 如果属性类型的一个属性或其子类型之一有一个等于所请求值的值，那么matched构件将有值TRUE。此外，matchedSubtype构件将出现，对匹配的子类型，将有值TRUE。如果没有任何匹配的属性类型或子类型，那么matched构件将有值FALSE；

– 如果返回一个错误，那么将采取failure选择方案。

8.6 添加操作

一个添加操作包括一个添加请求和一个对应的添加结果。

一个添加请求作为addRequest内容类型的一个实例来承载，添加结果作为addResult内容类型的一个实例来承载。

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

An instance of an **addRequest** content type is used to add a new object to the information system.

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

AddRequest数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有待添加新对象的唯一名称；

b) 当出现时，attr构件将持有一个或多个与新对象相关的属性。

如果访问者没有针对对象类的add许可，那么将返回一个insufficientAccessRight错误代码。

如果一个对象已经以提供的唯一名称存在，且访问者有discloseOnError许可，那么将返回一个objectAlreadyExists错误代码。否则，将返回一个insufficientAccessRight错误代码。

如果访问者针对随对象纳入的所有属性都没有add许可，那么请求失败。如果访问者针对所列的所有属性类型都有discloseOnError许可，那么将返回一个insufficientAccessRight错误代码。否则，将返回一个noInformation错误代码。

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

The privilege verifier shall use an instance of the **addResult** content type to return either the requested information or to report an error situation.

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

AddResult数据类型规定实际内容的语法，包括下列构件：

a) 如果添加了对象，那么将采取success选择方案；

b) 如果返回一个错误，那么将采取failure选择方案。

8.7 删除操作

一个删除操作包括一个删除请求和一个对应的删除结果。

一个删除请求作为deleteRequest内容类型的一个实例来承载，删除结果作为deleteResult内容类型的一个实例来承载。

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

An instance of a **deleteRequest** content type is used to delete an existing object from the information system.

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

DeleteRequest数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有待删除条目的唯一名称。

如果要删除的对象不存在，那么将返回一个noSuchObject错误代码。

如果访问者没有针对对象类的delete许可，那么若访问者有针对对象的discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchObject错误代码。

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

权限验证者将使用deleteResult内容类型的一个实例来返回所请求的信息或报告错误情况。

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

DeleteResult数据类型的一个实例有两种选择方案：

a) 如果对对象做了删除，那么将采取success选择方案。

b) 如果报告一个错误，那么将采取failure选择方案。

8.8 修改操作

一个修改操作包括一个修改请求和一个对应的修改结果。

一个修改请求作为modifyRequest内容类型的一个实例来承载，修改结果作为modifyResult内容类型的一个实例来承载。

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

使用**modifyRequest**内容类型的一个实例来修改一个现有的对象。

```
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}},
  ... }
```

ModifyRequest数据类型规定实际内容的语法，包括下列构件：

a) object构件将持有待修改对象的唯一名称：
   - 如果对象不存在，那么将返回一个noSuchObject错误代码；
   - 如果访问者没有针对对象的modify许可，那么若访问者有针对对象的discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchObject错误代码。

b) changes构件将持有有关修改一个或多个属性的信息：
   - addAttribute选择方案将持有一个待添加的新属性：
     i) 如果访问者没有针对属性类型的add许可，那么返回一个insufficientAccessRight错误代码；
     ii) 如果已经存在一个类型属性，那么若访问者有针对属性类型的discloseOnError许可，则返回一个attributeAlreadyExists错误代码。否则，将返回一个insufficientAccessRight错误代码；
   - deleteAttribute选择方案将确定要删除的属性：
     i) 如果访问者没有针对属性类型的delete许可，那么将返回一个insufficientAccessRight错误代码：
     ii) 如果不存在一个类型属性，那么将返回一个noSuchAttribute错误代码；
   - addValues选择方案将通过一个属性类型来确定现有的属性。有待添加至属性的各值为包含在该选择方案中的值。
     i) 如果对象未持有给定类型的任何属性，那么若访问者有discloseOnError许可，则返回一个noSuchAttribute错误代码。否则，将返回一个insufficientAccessRight错误代码；
     i) 如果访问者没有addValue许可，那么将返回一个insufficientAccessRight错误代码；
     ii) 如果试图添加一个已经存在的值，那么若访问者有discloseOnError许可，则返回一个attributeValueAlreadyExists错误代码。否则，将返回一个insufficientAccessRight错误代码；
   - 该deleteValues选择方案将通过属性类型来确定一个属性。有待从属性中删除的各值为包含在该选择方案中的值。
     i) 如果对象未持有给定类型的任何属性，那么若访问者有discloseOnError许可，则返回一个noSuchAttribute错误代码。否则，将返回一个insufficientAccessRight错误代码；
     ii) 如果访问者没有deleteValue许可，那么若访问者有discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchAttributeValue错误代码；

iii) 如果访问者试图删除一个不存在的属性值，那么将返回一个noSuchAttributeValue错误代码；

– replaceAttribute选择方案将用相同类型的一个新属性来替代现有的属性。

i) 如果对象未持有给定类型的任何属性，那么若访问者有discloseOnError许可，则返回一个noSuchAttribute错误代码。否则，将返回一个insufficientAccessRight错误代码；

ii) 如果访问者没有replaceAttribute许可，那么若访问者与discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchAttribute错误代码。

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

The privilege verifier shall use an instance of the **modifyResult** content type either to return the requested information or to report an error situation.

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

ModifyResult数据类型的一个实例有两种选择方案：

a) 如果对对象做了修改，那么将采取success选择方案。

b) 如果返回一个错误，那么将采取failure选择方案。

8.9 重命名对象操作

一个重命名对象操作包括一个重命名请求和一个对应的重命名结果。

一个重命名请求作为renameRequest内容类型的一个实例来承载，重命名结果作为renameResult内容类型的一个实例来承载。

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

An instance of a **renameRequest** content type is used to change the name of an existing object.

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

RenameRequest数据类型规定实际内容的语法，包括下列构件：

a) object构件将规定待重名对象当前的唯一名称；

b) new构件将为对象提供新的唯一名称。

如果待重命名对象不存在，那么将返回一个noSuchObject错误代码。

如果访问者没有针对命名对象的rename许可，那么若访问者有针对命名对象的discloseOnError许可，则返回一个insufficientAccessRight错误代码。否则，将返回一个noSuchObject错误代码。

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

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

RenameResult数据类型的一个实例有两种选择方案：

a) 如果对对象做了修改，那么将采取success选择方案。

b) 如果返回一个错误，那么将采取failure选择方案。

8.10 错误处置

在处置一个请求期间，若触发一个异常状况条件，则接收方必须通过在结果中纳入AccessErr数据类型的一个实例来返回一个错误代码。

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

如果在评估CMS定义的内容类型时触发一个异常状况条件，那么将采取cmsErr选择方案（参见A.5节）。

如果在评估CMS内容类型的实例时未检测到任何错误，但在PBACT特定内容类型的一个封装实例中检测到了一个错误，那么将采取pbactErr选择方案。

8.11 信息选择

InformationSelection数据类型用于规定一个读或修改请求可以请求什么信息。

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

InformationSelection数据类型有下列构件：

a) attributes构件将规定应返回什么属性。它有两种选择方案：

– 如果访问者想要关于对象的所有信息，那么将采用allAttributes选择方案；或者

– 如果只请求一组选定的属性，那么将采用select选择方案；

b) infotype构件有下列枚举项：

– 如果只需返回属性类型，那么将取attributeTypesOnly枚举项。在这种情况下，依据当前权限，访问者有针对属性类型的读许可。如果情况并非如此，那么从结果中删去属性类型。如果这导致没有任何信息要返回，那么请求失败；

– 如果对有效的权限既返回类型也返回值，那么将取attributeTypesAndValues枚举项。在这种情况下，依据当前权限，访问者有针对属性类型的读许可。如果情况并非如此，那么从结果中删去属性类型和值。如果这导致没有任何信息要返回，那么请求失败。

8.12 对象信息

当要返回关于一个对象的信息时，它将作为下列数据类型的一个实例来返回：

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

object构件将持有对象的唯一名称，信息返回给该对象。

info构件将持有一组属性，这些属性持有所请求的信息或属性类型集。

如果没有任何信息要返回，那么请求将失败。

8.13 定义的错误代码

有关特定PBACT内容类型的错误代码定义如下：

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

a) 如果访问者规定了一项服务，而它对之没有任何许可，并且不允许或不支持了解任何有关情况，那么将返回noSuchService错误代码；

b) 如果请求的操作与请求的服务不相关，那么将返回invalidOperationForService错误代码；

c) 当访问者请求服务时，而它对之没有任何许可，或者当它想对之执行一个操作时，对要访问的服务没有任何许可，那么将返回insufficientAccessRight错误代码；

d) 如果访问者试图访问一个并不存在的对象，或者访问一个不允许了解其是否存在的对象，那么将返回noSuchObject错误代码；

e) 如果访问者试图访问一个并不存在的属性，或者访问一个不允许了解其是否存在的属性，那么将返回noSuchAttribute错误代码；

f) 如果访问者试图访问一个并不存在的属性值，或者访问一个不允许了解其是否存在的属性值，那么将返回noSuchAttributeValue错误代码；

g) 如果试图为某个对象添加一个等同于已存在对象之唯一名称的唯一名称（条件是允许访问者了解该对象是否存在），那么将返回objectAlreadyExists错误代码；

h) 如果试图为某个已有类型属性的对象再添加一个属性（条件是允许访问者了解对象中该属性是否存在），那么将返回attributeAlreadyExists错误代码；

i) 如果试图为某个已有相同类型属性的对象再添加一个属性（条件是允许访问者了解对象中该属性是否存在），那么将返回attributeValueAlreadyExists错误代码；

j) 如果请求信息，但无法使用或者不允许访问者了解该数据是否存在，那么将返回noInformation错误代码。

9 权限分配协议

9.1 协议范围

权限分配协议用于分配权限：

a) 从一个机构源（SOA）到一个中间属性机构（AA），以便进一步授权；

b) 从一个SOA直接到一个实体，使用已分配的权限来声明这些权限；

c) 从一个AA直接到一个实体，使用已分配的权限来声明这些这些权限；以及

d) 当有多个AA处于SOA与权限持有者之间的路径上时，从一个AA到另一个 AA。

注 — 建议授权路径尽可能短。

9.2 内容类型

权限分配协议利用两种内容类型，一种内容类型用于分配权限，一种内容类型用于确认分配情况。

9.2.1 权限分配请求内容类型

在一个privAssignRequest内容类型的实例中承载一个权限分配请求。

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

实际内容的语法通过下列数据类型来规定：

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

该数据类型只有一个持有属性证书序列的构件。如果权限分配请求发送自某个SOA，那么序列将只包含一个属性证书。如果在SOA与权限持有者之间存在一个单一AA，那么从AA到权限持有者的请求将既包括由SOA发放的属性证书，又包括由AA发放的属性证书。对SOA与权限持有者之间的每个额外的AA，都需要多个属性证书。

9.2.2 权限分配结果内容类型

在privAssignResult内容类型的一个实例中承载的一个权限分配结果。

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

实际内容的语法通过下列数据类型来规定：

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

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

9.2.3 定义的错误代码

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

附件A

关于ITU-T 1080系列的对象标识符分配

（本附件构成本建议书的组成部分）

A.1 对象标识符树的顶层

[ITU-T X.1081]附件A在分配给电子生物特征的拱之下分配弧，它是：

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

在该弧下，[ITU-T 1081]为远程医疗分配下列弧：

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

[ITU-T X.1080.1]在**id-th**弧下分配了若干弧。弧"0"分配给在[ITU-T 1080.1]中定义的ASN.1模块。其他弧分配给各实体类别。本建议书规定，弧"0"通常也应用于分配ITU-T X.1080系列中的对象标识符。为了避免冲突，值10用于分配ITU-T X.1080系列中的对象标识符。

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

下列弧分配给ITU-T X.1080系列的不同部分：

```
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) }
----
```

ITU-T X.1080系列某个特定部分的一个弧划分如下：

– 各模块有弧"0"；

– CMS内容类型有弧"1"；

– 属性类型有弧"2"。

某个特定部分可依据其需求分配额外的弧。

对本建议书，下列弧分配给各模块：

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

下列弧分配给CMS内容类型：

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

下列弧分配给属性类型，用于分配权限：

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

A.2 CMS内容类型的对象标识符

下列对象标识符分配给内容类型，为权限分配协议和权限评估协议而定义：

```
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 权限属性类型的对象标识符

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

附件B

加密消息语法概要文件

（本附件构成本建议书的组成部分）

B.1 概述

加密消息语法（CMS）在[IETF RFC 5652]中进行定义。它定义了有关数据完整性、认证和机密性的通信能力。[IETF RFC 5083]添加了额外的规范。[IETF RFC 5911]和[IETF RFC 6268]为CMS提供了新的ASN.1。本附件参照这些规范，提供了一个有关CMS的概要文件，供电子生物特征规范使用。

CMS是一个通用的规范，可用在许多不同的环境中。本概要文件不利用所有的CMS功能。本概要文件包含本建议书中使用的CMS数据类型和其他的电子生物特征规范。这些其他的电子生物特征规范可参考本附件，以便使用CMS。

无意为某个实施方案制定一个不符合IETF RFC要求的规范，而只是对CMS的这些方面问题做一讨论，它们与电子生物特征规范有关。附录 I提供了一个非正式的ASN.1模块，以反映电子生物特征对CMS的使用情况。

CMS定义了不同的内容类型，可用于不同的目的。电子生物特征规范利用signedData内容类型来提供认证和完整性，它们利用envelopedData内容类型来提供加密，从而实现保密，它们也利用ct-authEnvelopedData内容类型。当需要数字签名时，使用signedData内容类型，当需要保密时，使用envelopedData内容类型。使用时，envelopedData内容类型的一个实例被封装在signedData内容类型的一个实例中。当多个消息构成一个特定的任务时，使用ct-authEnvelopedData内容类型。

电子生物特征规范并不会用到上述内容类型的所有方面。因此，本附件只提供了一个关于在电子生物特征中如何使用CMS的概要文件。为方便参考起见，此处包含了CMS的相关各方面问题。

如果需要保密，那么电子生物特征规范定义的内容类型的一个实例将被封装在envelopedData内容类型的一个实例中；否则，它将被封装在signedData内容类型的一个实例中。作为选择，这样一个内容类型实例也可包括在ct-authEnvelopedData内容类型的一个实例中。

使用下列信息对象类，依据[IETF RFC 6268]来定义一个内容类型：

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

CONTENT-TYPE信息对象类相当于ASN.1建立信息对象类TYPE-IDENTIFIER。一个CONTENT-TYPE信息对象用于将由一个对象标识符确定的内容类型绑定于内容的抽象语法。

下列ContentInfo数据类型提供了有关内容类型的通用语法：

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

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

支持的内容类型是signedData内容类型、envelopedData内容类型、ct-authEnvelopedData内容类型以及通过某个特定的电子生物特征规范定义的内容类型集合。

CMS要求应为一个数据类型规定一个CMS版本，以指明该数据类型所用的特定语法。定义了下列版本：

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

[IETF RFC 6268]定义了下列参数化的数据类型，用于整个规范：

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

B.2 signedData内容类型的使用

下列内容类型在[IETF RFC 5652]第5节中予以规定。使用一个稍经修改的记法，反映其在电子生物特征技术中的用法，它规定如下：

```
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,
  ... }
```

注1 — [IETF RFC 6268]使用AlgorithmIdentifier数据类型某个经适当修改的版本。不过，本概要文件使用在[ITU-T X.509]中定义的AlgorithmIdentifier数据类型。

依据[IETF RFC 5652]第5.1节，version构件应取值v3。

digestAlgorithms构件将由单个元素组成，自通过适用的电子生物特征规范定义的、适用的哈希算法集中规定一个哈希算法。

下列定义允许纳入任何哈希算法。

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

注2 — 本概要文件不强制要求使用某个特定的哈希算法。参考规范或实施者的协议可用一个在某个特定环境中支持的哈希算法集来替代各点。

注3 — CMS允许多重数字签名，并因此允许多重哈希算法。多重数字签名不与电子生物特征规范相关。

encapContentInfo构件将持有下列数据类型的一个实例：

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

本数据类型有下列构件：

a) eContentType构件将持有对象标识符，用于确定封装的内容类型。如果需要加密，那么本构件将持有envelopedData内容类型的身份。如果不需要加密，那么本构件将持有其中的一个内容类型，它通过相关的电子生物特征规范来规定；

b) econtent构件将持有封装在一个八进制字符串中的、实际的封装内容。它将始终存在。

注4 — 本构件定义为是可选的。不过，所有相关的内容类型都有确切的内容。

certificates构件将持有足以建立单个认证路径的公开密钥证书，如[ITU-T X.509]中定义的PkiPath数据类型所规定的那样。

crls构件与电子生物特征规范不相关，并将不存在。

signerInfos构件将持有SignerInfos数据类型的一个实例：

```
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 ::= {...}
```

本概要文件仅支持一个签名者，因此SignerInfos数据类型的一个实例有且只有一个元素。SignerInfo数据类型有下列构件：

a) version构件将依据[IETF RFC 5652]来规定v1；

b) sid构件将确定签名者的终端实体公开密钥证书，并持有SignerIdentifier数据类型的一个实例。本数据类型规定了两个选择方案；

   – issuerAndSerialNumber选择方案通过规定发放证书之CA的唯一名称和公开密钥证书的序列号，来确定终端实体公开密钥证书。将总采用本选择方案；

   – 将不采用subjectKeyIdentifier选择方案；

c) digestAlgorithm构件将具有与SignerInfo数据类型的digestAlgorithms构件中所用的相同值；

d) signedAttrs构件将持有一个已签署属性的清单。[IETF RFC 5652]需要至少应包括contentType和messageDigest属性类型的实例。本概要文件不要求包括任何额外的属性，但可将参考规范添加至清单中；

e) signatureAlgorithm构件将持有签名算法，用于创建签名构件持有的数字签名。

注5 — 本概要文件不强制要求使用某个特定的签名算法。参考规范或实施者的协议可用一个用于支持某个特定电子生物特征规范的签名算法集来替代各点。

f) signature和unsignedAttrs构件如[IETF RFC 5652]所规定。

B.3 envelopedData内容类型的使用

B.3.1 概述

envelopedData内容类型允许对数据进行加密。这需要建立一个共享的、对称的密钥。[IETF RFC 5652]提供了不同的技术，用于生成此类对称的密钥。本概要文件需要称为Diffie-Hellman（DH）密钥协议方法的密钥协议技术。Diffie-Hellman（DH）密钥协议方法由[IETF RFC 2631]为非椭圆曲线技术而定义。[IETF RFC 5753]提供了使用椭圆曲线技术的规范。

DH方法的结果是产生一个共享密钥，它可用作密钥生成材料，允许产生共享的、对称的密钥。本概要文件认可两种DH操作模式：短暂-静态模式和静态-静态模式。

短暂-静态模式要求接收者拥有一个带DH公开密钥的公开密钥证书，它经过发放CA的认证。对发送者而言，该公开密钥证书应是可用的。发送者为其发送的每一条消息创建一个新的DH密钥对。通过这种方式，对每一条消息而言，共享密钥都将变得不同。

静态-静态模式要求每一方通信实体都有一个经过认证的DH公开密钥证书。由于该模式将为每一条消息产生相同的共享密钥，因此发送者需要提供一些随机的用户密钥生成材料，以便为每一条消息获得不同的密钥生成材料。

本概要文件要求应支持短暂-静态模式。

这两种方法都要求在两个方向上的通信进行过程中，通信中的两个实体都拥有其合作伙伴的、经认证的DH终端实体公开密钥证书。

下列内容类型在[IETF RFC 5652]第6节中予以规定，并通过[IETF RFC 6268]进行更新：

```
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 }
```

EnvelopedData构件有下列构件：

a) 如果unprotectedAttrs构件存在，那么依据[IETF RFC 5652]，version构件将拥有值v2；否则，它应为值v0。

b) originatorInfo构件将不存在。

c) recipientInfos构件将持有RecipientInfos数据类型的一个实例，如B.3.2节中所规定的那样。

d) encryptedContentInfo构件将持有EncryptedContentInfo数据类型的一个实例，如B.3.4节中所规定的那样。

e) 如果希望将要在所议方向上进行传输的下一条消息是ct-authEnvelopedData内容类型的一个实例，那么需要unprotectedAttrs构件；否则，它可不存在。

B.3.2 接收者信息

RecipientInfos数据类型允许有多个RecipientInfo数据类型的实例。不过，出于本概要文件的目的，它将被限于只有单个实例。RecipientInfo数据类型的一个实例规定关于如何在两个通信实体之间建立一个共享密钥的不同选择方案。

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

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

本概要文件只使用了两个RecipientInfo选择方案，它们在[IETF RFC 5652]中予以定义：

a) kari选择方案是用于envelopedData内容类型的唯一选择方案。KeyAgreeRecipientInfo数据类型提供所需的信息，以建立一个共享密钥，如B.3.3节详述；

b) kekri选择方案是用于ct-authEnvelopedData内容类型的唯一选择方案。KEKRecipientInfo数据类型提供所需的信息，以建立一个共享密钥，如B.4节详述。

B.3.3 密钥协议

```
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
```

KeyAgreeRecipientInfo数据类型有下列构件：

a) 依据[IETF RFC 5652]，version构件将取值v3；

b) originator构件将以下列选择方案持有OriginatorIdentifierOrKey数据类型的一个实例：

- 如果使用静态-静态方法，那么将采取issuerAndSerialNumber选择方案；然后将持有IssuerAndSerialNumber数据类型的一个实例。本数据类型将确定发送者的DH公开密钥证书。
  - i) issuer构件将持有发放CA的唯一名称，并等同于所议之公开密钥证书的issuer构件；
  - ii) serialNumber构件将等同于所议之公开密钥证书的serialNumber构件。
- 将不采用subjectKeyIdentifier选择方案。[参见附件B、B.2、最后一个b)、最后一点]；
- 对短暂-静态Diffie-Hellman方法，将采取originatorKey选择方案；然后将以下列构件持有OriginatorPublicKey数据类型的一个实例：
  - i) algorithm构件将参考所用的Diffie-Hellman公开密钥算法；

  注1 — 本概要文件不强制要求使用某个特定的Diffie-Hellman公开密钥算法。参考规范或实施者的协议可用一个在某个特定环境中支持的Diffie-Hellman公开密钥算法集来替代各点。
  - ii) publicKey构件将持有发送者生成的Diffie-Hellman公开密钥。发送者将为该内容类型的每次使用生成一个新的Diffie-Hellman密钥对。

从本地的私有密钥和接收者的公开密钥，发送者可生成共享密钥。接收者将利用其私有密钥连同在OriginatorPublicKey或IssuerAndSerialNumber数据类型中提供的发送者的公开密钥生成一个完全相同的共享密钥。

c) 当采用静态-静态方法时，ukm构件将出现，并将持有UserKeyingMaterial数据类型的一个实例。

从共享密钥，ukm构件中的值（如果相关的话）和其他一些在[IETF RFC 2631]中规定的信息，双方将产生所谓的密钥加密密钥（KEK）。随后，该密钥用于加密由发送者产生的内容加密密钥（CEK）。这种技术被称为密钥封装；

d) keyEncryptionAlgorithm 构件将规定密钥封装算法，并将持有KeyEncryptionAlgorithmIdentifier数据类型的一个实例；

注2 — 本概要文件不强制要求某个特定的密钥封装算法集。可在未来定义新的算法。高级加密标准（AES）密钥封装算法由[IETF RFC 3394]定义。参考规范或实施者的协议可用一个在某个特定环境中支持的封装算法集来替代各点。

e) recipientEncryptedKeys构件将持有RecipientEncryptedKeys数据类型的一个实例。这样的一个实例将只由单个元素组成，即RecipientEncryptedKey数据类型的单个实例。该数据类型有下列两个构件：

– rid构件将通过其终端实体公开密钥证书持有接收者的身份证明；

– encryptedKey构件将持有经过加密的CEK，用于内容的加密，如c)项中所讨论的那样。

B.3.4 CMS内容加密密钥的重用

如果CEK将用于后续的ct-authEnvelopedData内容类型实例，如[IETF RFC 3185]中所规定的那样，那么适当的参考信息将进入未经保护的属性中，如B.3.1节中所讨论的那样。该信息将由双方留存。如果需要高水平的安全，那么类型的属性aa-CEKMaxDecrypts 应为值“1”或被省略。

B.3.5 加密的内容信息

EncryptedContentInfo数据类型的一个实例持有经过加密的封装内容。

```
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
```

EnvelopedData数据类型的encryptedContentInfo构件将持有EncryptedContentInfo数据类型的一个实例：

a) contentType构件将持有加密内容的内容类型。可能的内容类型的清单针对的是那些加密是一种选项的内容类型。

b) contentEncryptionAlgorithm和encryptedContent构件，如[IETF RFC 5652]所要求的那样。

B.4 经过认证的-经过封装的-数据内容类型的使用

B.4.1 概述

如[ITU-T X.1080.1]规定，有关电子生物特征的协议一般包括设置交换，以建立一个会话，后跟多个信息交换，并以会话终止来结束。在这样一个环境中，它可能不需要为每条消息建立一个新的密钥加密密钥。

[IETF RFC 5083]规定了一个内容类型ct-authEnvelopedData，它不包括在[IETF RFC 5652]中。该内容类型允许使用高效的认证加密技术。本概要文件利用[IETF RFC 5084]中定义的AES-GCM算法，连同重用[IETF RFC 3185]中定义的CEK。关于细节，应参看这些规范。

```
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 }
```

AuthEnvelopedData数据类型有下列构件：

a) 依据[IETF RFC 5083]，version构件将取值v0；

b) originatorInfo构件将不出现；

c) recipientInfos构件将持有RecipientInfos数据类型的一个实例。对该数据类型，在B.3.2节中进行描述。kekri选择方案，除了kari选择方案，与此内容类型相关。当采用kekri选择方案时，该选择方案将持有KEKRecipientInfo数据类型的一个实例，如B.4.2节所规定的那样；

d) authEncryptedContentInfo构件将持有EncryptedContentInfo数据类型的一个实例，如B.3.5节所规定的那样；

e) authAttrs构件，当出现时，将持有一组受认证保护的属性；

f) mac构件将持有生成的消息认证码（MAC）；

g) unauthAttrs构件将持有相同类型的属性，如B.3.1 节e)项中所规定的那样。如果知道内容类型实例是所议方向上会话的最后一个实例，那么该构件可不出现。

B.4.2 KEK接收者信息

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

```
  encryptedKey            EncryptedKey }

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

KEKRecipientInfo数据类型有下列构件：

a) 依据[IETF RFC 5652]，version构件将取值v4；

b) kekid构件将以下列构件持有KEKIdentifier数据类型的一个实例：

   – keyIdentifier构件将持有有关CEK的标识符，保留自先前的交换，如B.3.4节所规定的那样；

   – 不需要其他的构件；

c) keyEncryptionAlgorithm构件将持有封装算法，如B.3.3节条目d)所规定的那样。对有关某个特定电子生物特征会话的所有内容实例，建议使用相同的封装算法。

B.5 属性

下列属性类型通过[IETF RFC 5652]来定义。这些属性类型的实例将作为有符号属性纳入。

```
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 }
```

下列属性类型通过[IETF RFC 3185]来定义。这些属性类型的实例可以作为无符号属性纳入。

```
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 加密消息语法错误代码

出于CMS所有可能的用途，[b-IETF RFC 7191]提供了一份关于CMS错误代码的清单。下面列出的是有关电子生物特征的、这些错误代码的一个子集。有关错误代码的描述，请查阅[b-IETF RFC 7191]。

当[b-IETF RFC 7191]指的是一个证书时，它是公开密钥证书的一个参照，用于CMS定义的内容类型。

```
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) }
```

附件C

关于权限声明和分配协议的正式规范

（本附件构成本建议书的组成部分）

```
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

  -- from Rec. ITU-T X.501 | ISO/IEC 9594-2

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

  -- from Rec. ITU-T X.509 | ISO/IEC 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,
  insufficientAccessRight,
  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),
  ... }

-- object identifier allocations
```

```
-- top tree

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) }

-- Content types

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) }


-- Artribute types for carring privilege definitions

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

END
```

附录I

关于加密消息语法概要文件的非正式规范

（本附录不构成本建议书的组成部分）

一个只支持CmsTelebiometric模块的实施方案不符合IETF CMS规范的要求，不得被视为一个实施方案规范。在此仅作为参考资料并供一致性检查。

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

-- EXPORTS All

IMPORTS

  -- from Rec. ITU-T X.501 | ISO/IEC 9594-2

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

  -- from Rec. ITU-T X.509 | ISO/IEC 9594-8

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

  -- from Rec. ITU-T X.520 | ISO/IEC 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,
--date          GeneralizedTime OPTIONAL,
--other         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
```

参考资料

[b-ITU-T X.841] ITU-T X.841建议书 (2000) | ISO/IEC 15816:2002，信息技术 – 安全技术 – 用于访问控制的安全信息对象。

[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.

ITU-T 系列建议书

A系列	ITU-T工作的组织
D系列	一般资费原则
E系列	综合网络运行、电话业务、业务运行和人为因素
F系列	非话电信业务
G系列	传输系统和媒质、数字系统和网络
H系列	视听及多媒体系统
I系列	综合业务数字网
J系列	有线网络和电视、声音节目及其它多媒体信号的传输
K系列	干扰的防护
L系列	电缆和外部设备其它组件的结构、安装和保护
M系列	电信管理，包括TMN和网络维护
N系列	维护：国际声音节目和电视传输电路
O系列	测量设备的技术规范
P系列	电话传输质量、电话设施及本地线路网络
Q系列	交换和信令
R系列	电报传输
S系列	电报业务终端设备
T系列	远程信息处理业务的终端设备
U系列	电报交换
V系列	电话网上的数据通信
X系列	**数据网、开放系统通信和安全性**
Y系列	全球信息基础设施、互联网协议问题和下一代网络
Z系列	用于电信系统的语言和一般软件问题

瑞士印刷
2017年，日内瓦