ICS 35.100.70
L 79

中华人民共和国国家标准

GB/T 19771—2005

信息技术　安全技术　公钥基础设施 PKI组件最小互操作规范

Information technology—Security technology—Public key infrastructure —Minimum interoperability specification for PKI components

2005-05-25 发布　　2005-12-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　　次

前　言

本标准是在参考美国国家标准与技术研究院(NIST)提出的《公钥基础设施 PKI 组件最小互操作规范》第二版内容的基础上修改而成，同时本标准还参照了包括证书管理策略(CMP)、证书请求消息格式(CRMF)、FIPS 许可的密码算法和 X9 密码算法等相关的规范。

本标准凡涉及密码算法相关内容，按国家有关法规实施。

本标准中引用的 SHA-1、RSA、SHA1-MAC、SHA1-HMAC 、DES-MAC、tDEA 密码算法均为举例性说明，具体使用时均须采用国家商用密码管理委员会批准的相应算法。

本标准的附录 A、附录 B、附录 C、附录 D 为规范性附录。

本标准由中华人民共和国信息产业部提出。

本标准由全国信息安全标准化技术委员会(TC260)归口。

本标准起草单位：信息安全国家重点实验室、中国电子技术标准化研究所。

本标准主要起草人：冯登国、吴志刚、荆继武、高 能、向 继、张 凯、周瑞辉、徐 佳、林璟锵、曹 政、余婧、廖洪銮、李 丹、罗锋盈、陈 星。

引　言

数字签名证书在政府服务商业和法律程序中代替手写签名，并且允许以前没有联系的双方可靠地鉴别对方以进行商业事务。加密证书提供了加密传输和加密算法的应用，来建立或保护对称密钥以提供机密性。这样的一个公钥基础设施（PKI）系统和它相应的证书，也许远远超出了一些应用的实际需要，对那些特别的应用要求来说改进的证书和协议更合适。

信息技术　安全技术　公钥基础设施 PKI组件最小互操作规范

1　范围

本标准支持大规模公钥基础设施(PKI负责发布、撤销和管理用于数字签名及密钥管理的公钥证书)的互操作性。本标准为不同的PKI开发者所开发的组件产品提供了基本的互操作性参考。

本标准的内容涉及：

- 公钥证书的产生、更新和撤销；
- 签名的产生和验证；
- 证书和证书认证路径验证。

本标准主要包括了对证书、证书撤销列表(CRL)扩展和一套事务的描述。这些事务包括证书申请、证书更新、证书撤销以及从资料库检索证书和CRL。

本标准主要以最终用户的角度来看待PKI的互操作性，即怎样申请和获得一个证书；怎样签署文档；怎样检索他人的证书；怎样验证签名。就像下面所提及的，PKI的“内部”操作规范还没有达到足够成熟，因此它们没有被详细规定。

在本标准中PKI被分成五个组件：

- 颁发和撤销证书的证书认证机构(CAs)；
- 确保公钥和证书持有者的身份以及别的属性之间绑定的注册机构(RAs)；
- 获得证书和签署文档的证书持有者；
- 验证签名并且执行密钥管理协议以及验证证书认证路径的客户；
- 存储并提供对证书和CRL查询的资料库。

许多实体在功能上既是证书持有者又是客户。CAs和RAs也是如此。终端实体证书持有者通常也是客户。当然，也有一些客户并不是证书持有者。

资料库不必是证书持有者和客户。本标准仅仅涉及资料库协议的一部分，那就是客户要求从资料库中获得证书和CRL的信息。

本标准将轻型目录访问协议(LDAP)版本2作为用户访问资料库的传输手段，因为它是被广泛接受和采用的方法。例如，这种选择既不强调CA用来更新资料库的标准化协议，也不强调资料库之间互相映射的协议，尽管它们都是需要的。前者可以具体情况具体分析以解决CA和资料库之间的协议，后者也许并不必要。

在通常的证书状态确认(本标准遵循的)中，资料库不是可信实体，CA对CRL的签名更可靠。在线证书状态实时确认机制要求资料库是可信实体，而且它们也能让客户相信他们的身份。这样的证书状态确认协议超出了本标准的范围，但是在一些应用中可能需要实时证书状态确认，所以在以后的修订版中可能会解决这个问题。

本标准中没有提供让资料库验证使用者的协议，该协议是资料库记费应用的前提。虽然这可能是资料库重要的商用模式，但目前人们对该模式的看法还没有达到一致，也没有统一的支撑协议。在以后的修订版中可能会解决这个问题。

在一些情况下，带外事务也是本标准中事务的一部分。带外事务的形式和内容超出了本标准的范围。

本标准假定CA、RA和证书持有者是物理上分离的。如果这些实体在物理上是在一起的话，那么

对特定接口的支持是不需要的。具体地说，如果一个 PKI 组件既包含 RA 又包含 CA 的功能，那么就不必支持这两者之间的事务消息格式。然而，如果一个系统包括一个 CA，该 CA 除了具有本地 RA 功能以外还支持远程 RA，那么它就必须支持和远程 RA 之间的事务。在以下的论述中，我们假设 CA 和 RA 是单独的 PKI 组件。

本标准把 CA 和 RA 当作 PKI 系统的功能实体。这些实体的内部设计超出了本标准的范围。

本标准假设，从最小范围来讲，证书持有者有一个签名密钥和证书。可选的，证书持有者还可以获得一个加密密钥和证书。一旦证书持有者希望请求或者撤销加密证书，它就需要用签名密钥来向 CA 证明自己。

对那些没有签名密钥对、不需要不可否认性服务的系统，本标准不予直接支持。当然，这些实体主要是这样一些计算机系统(例如，路由器或是链路加密机)，它们由管理员来维护。如果管理员有系统管理的签名密钥对，本标准的事务集合也可用来支持这些实体的证书请求和撤销。

本标准选定了一个成熟重要的数字签名算法，新的标准算法很容易加入进来。

本标准支持层状和网状信任模型。在层次模型中，CA 通过认证一个次级 CA 来提供可信性。信任授权从根 CA 开始，该 CA 被所有节点信任。在网状模型中，信任是建立在两个同等关系的 CA 中的(即交叉认证)，因此两个 CA 之间可以有多个信任路径。最小互操作规范假设 GB/T 16264.8—2005 证书的扩展 basicConstraints，nameConstraints，keyUsage 和 certificatePolicy 都会包含在证书中，以便对信任关系进行明确管理。

本标准假设无需确认就可以从资料库中检索证书和 CRL。客户可以从适当的资料库中获得证书和 CRL 来进行路径验证。资料库可以是 X.500 目录或者使用通用资源标识符(URI)可访问的目录。希望资料库能够支持 LDAP(即 RFC 1777)，因此相应的产品也要求支持这个协议。

资料库不必连接在一起，别的协议也可以用来获得证书和 CRL。本标准要求明确所使用的证书资料库和检索证书以及 CRL 的机制。

CRL 是一个广泛使用的机制，它用于撤销证书和验证未到期证书的状态。CRL 的使用可能还没有统一。一些 CA 选择在线实时确认证书状态的机制，CRL 的产生对于别的 CA 的使用者来说应该具有互操作性。除了当前检查证书的有效性，CRL 还提供了一个重要的机制，即将证书以前的撤销状态存档。如果一个带日期签名的签名日期在证书的有效期内，那么该签名是合法的，当前的 CRL 不会显示证书被撤销的信息。

在本标准中，假设 CA 可以产生 CRL，客户在验证证书时可以使用 CRL。

2 规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

GB/T 16264.8—2005 信息技术 开放系统互连 目录 第 8 部分：公钥和属性证书框架(ISO/IEC 9594-8:2001，IDT)

ISO/IEC 8825-1:2002 信息技术 ASN.1 编码规则 第 1 部分：基本编码规则(BER)、正则编码规则(CER)和非典型编码规则(DER)规范

ANSI X9.52 用于金融服务业的公钥密码算法：三重 DES 操作模式

ANSI X9.55 用于金融服务业的公钥密码算法：公钥证书扩展和证书撤销列表扩展

RFC 822 Internet 文本邮件的标准消息格式

RFC 1766 语言标识用标签

RFC 1777 轻量级目录访问协议

RFC 1959 LDAPURL 格式

RFC 2104　用于消息认证的带密钥散列函数
RFC 2202　HMAC-MD5 和 HMAC-SHA-1 测试用例
RFC 2313　PKCS#1　RSA 加密版本 1.5
RFC 2314　PKCS#10　认证请求语法版本 1.5
RFC 2459　因特网 X.509 公开密钥基础设施证书和证书撤销列表框架
RFC 2510　因特网 X.509 公开密钥基础设施证书管理协议
RFC 2511　因特网 X.509 公开密钥基础设施证书消息格式
RFC 2559　因特网 X.509 公开密钥基础设施操作协议-轻量目录访问协议版本 2
RFC 2985　PKCS#9　可选的对象类和参数类型版本 2.0
FIPS-113:1985　计算机数据加密鉴别
PKCS#11　密码令牌接口标准版本 2.0

3　术语和定义

下列术语和定义适用于本标准。

3.1

抽象语法记法一(ASN.1)　Abstract Syntax Notation 1(ASN.1)

用来组织复杂数据对象的表示法。

3.2

许可　accredit

认可一个实体或个人去执行特定动作。

3.3

公钥证书　public key certificate

用户的公钥连同其他信息,并由发布该证书的证书认证机构的私钥进行加密使其不可伪造。

3.4

证书持有者　certificate holder

有效证书的主体对应的实体。

3.5

证书策略　certificate policy

命名的一组规则,指出证书对具有公共安全要求的特定团体和/或应用的适用范围。例如,一个特定的证书策略表明,用于确认电子数据交换贸易证书的适用范围是价格在某一预定范围内的交易。

3.6

证书用户　certificate user

需要确切地知道另一实体的公开密钥的某一实体。

3.7

证书使用系统　certificate-using system

证书用户使用的、本标准定义的那些功能实现。

3.8

证书认证机构(CA)　Certificate Authority(CA)

负责创建和分配证书,受用户信任的权威机构。用户可以选择该机构为其创建密钥。

3.9

证书认证路径　certification path

一个 DIT 中对象证书的有序序列,通过处理该有序序列及其起始对象的公钥可以获得该路径的末端对象的公钥。

3.10

认证业务说明(CPS)　Certification Practice Statement(CPS)

证书认证机构发放证书时遵循的业务说明。

3.11

CRL 分布点　CRL distribution point

一个 CRL 目录项或其他 CRL 分发源;由 CRL 分布点分发的 CRL 可以包括仅对某 CA 所发证书全集某个子集的撤销条目,或者可以包括有多个 CA 的撤销条目。

3.12

证书撤销列表(CRL)　Certificate Revocation List(CRL)

一个已标识的列表,它指定了一套证书发布者认为无效的证书。除了普通 CRL 外,还定义了一些特殊的 CRL 类型用于覆盖特殊领域的 CRL。

3.13

发证　certify

颁发一个证书的行为。

3.14

客户　client

使用 PKI 来获得证书并且去验证证书和签名的功能。

3.15

增量 CRL　delta-CRL

部分撤销列表,在可参考的基础 CRL 发布以后,这些证书更改了其撤销状态。

3.16

可辨别编码规则(DER)　Distinguished Encoding Rules(DER)

对 ASN.1 对象进行编码的规则。

注:本标准中使用 DER 对 ASN.1 对象进行编码。

3.17

数字签名　digital signature

允许接收者验证签名人的身份和数据完整性的数据单元。

3.18

目录服务(DS)　Directory Service(DS)

分布在网络中的各种节点或服务器提供的分布式数据库服务。

3.19

终端实体　end entity

不以签署证书为目的而使用其私钥的证书主体或者是依赖(证书)方。

3.20

散列函数,哈希函数　hash function

将值从一个大的(可能很大)定义域映射到一个较小值域的(数学)函数。"好的"散列函数是把该函数应用到大的定义域中的若干值的(大)集合的结果可以均匀地(和随机地)被分布在该范围上。

3.21

散列码　hash code

散列函数的输出比特串。

3.22

消息认证码(MAC)　Message Authentication Code(MAC)

通过密码技术由消息产生的认证数据。

3.23

消息摘要　message digest

散列一个消息后得到的固定长度数据。

3.24

带外事务　out of band

不是通过电子形式，而是通过通常的物理形式进行的一些 PKI 组件的事务。

3.25

策略映射　policy mapping

当某个域中的一个 CA 认证另一个域中的一个 CA 时，在第二个域中的特定证书策略可能被第一个域中的证书认证机构认为等价(但不必在各方面均相同)于第一个域中认可的特定证书策略。

3.26

注册机构(RA)　Registration Authority(RA)

为用户办理证书申请、身份审核、证书下载、证书更新、证书注销以及密钥恢复等实际业务的办事机构或业务受理点。

3.27

资料库　repository

存储证书和 CRL 等信息，并提供无需验证的信息检索服务的数据库。

3.28

自颁发证书　self-issued certificate

证书的主体和颁发者相同的 CA 证书。

3.29

统一资源标识符(URI)　Uniform Resource Identifier(URI)

包含了名字或地址的短数据串，指向 web 上的某个对象。

3.30

统一资源定位符(URL)　Uniform Resource Locator(URL)

包含地址的短数据串，指向 web 上的某个对象，URL 是 URI 的子集。

4　缩略语

下列缩略语适用于本标准。

CA　证书认证机构

CRL　证书撤销列表

PKCS　公钥密码系统

PKI　公钥基础设施

POP　拥有证明

RA　注册机构

5　PKI 组件规范

5.1　概述

本章规定了 PKI 各组件进行互操作时所需的功能和事务的最小集合。它们分别是 CA、RA、证书持有者和 PKI 客户的规范。

5.2　证书认证机构(CA)

5.2.1　概述

CA 负责生成、撤销、公布和存档证书。资料库使得所有证书使用者都可以获得证书和 CRLs 的

信息。

CA 生成自己的公私钥对并公布自己的证书。因此,CA 应当生成、估定相应的参数以便生成/验证它们的签名。为了使新的 CA 能加入到已有的层次结构中,它们应当可以从父 CA 那里申请证书。CA 也应可以生成交叉证书,在其他 CA 的策略允许下支持与其他 CA 进行交叉认证。CA 对所有的事务进行存档,这些事务包括 PKI 各组件之间的服务请求与响应。

CA 授权 RA 去确认那些申请证书的使用者的身份或其他的特征属性。这种授权通过离线接受来自某个 RA 的证书请求完成。CA 利用 X.500 的可辨别名(DN)来唯一标识证书持有者。

CA 本身也具备证书持有者的功能:请求、撤销、更新由其他 CA 颁发的证书(5.4);检索证书和 CRLs,验证证书认证路径的客户功能(5.5)。

5.2.2 与互操作性有关的 CA 功能要求

CA 执行下列功能:

- 颁发并传送证书给终端实体和其他的 CA;
- 接收来自 RA 的证书撤销请求;
- 将证书和 CRL 存入资料库;
- 请求 CA 证书。

a) 颁发数字签名证书

CA 支持三种有关数字签名证书的证书请求:RA 发起的注册请求、更新和自我注册请求。根据每种请求的不同,CA 以不同的方式来鉴别这些申请证书主体的身份。潜在的证书持有者在自我注册请求中提供一个证据证明自己的身份已经得到了 RA 的核实,该证据是由从 RA 那里获得的秘密信息导出的。当用户与 RA 物理上在一起时,RA 产生并签署基于 RA 的登记请求来保证该用户的身份。在证书更新请求中,当前有效证书的主体可以用它们的私钥签名来保证它们身份的真实性。

RA 发起的证书请求:RA 保证该用户的身份并将其和公钥绑定在一起。当 CA 接收到来自授权 RA 的证书请求时,CA 就处理该请求,如果接受,就生成新证书,并将其放到资料库中,然后将证书发给相应的 RA。CA 也可以直接把新证书发给该证书的持有者。如果基于 RA 的证书请求不是来自一个授权的 RA,即签名无效,或者包含不匹配信息,CA 将会拒绝该证书请求。如果 CA 拒绝了该证书请求,它将会向 RA 报告失败并说明原因。

在自我注册证书请求中,RA 为潜在证书持有者提供了一个秘密信息。请求实体产生自己的公私钥对,组成一个证书请求消息,并用相应的私钥签名,被签名部分包括基于 RA 提供的秘密导出的认证信息。CA 接收来自实体的证书请求,通过认证信息验证请求者的身份并且验证实体拥有相应的私钥。如果接受,CA 就产生新证书,并将其放到资料库中,然后将证书发给证书持有者。如果认证信息验证不通过,签名无效,或者包含不匹配信息,CA 将会拒绝该证书请求。如果 CA 拒绝了自我注册证书请求,它将会向申请者报告失败并说明原因。

证书更新请求:申请者已确定的身份通过该更新请求消息来验证。证书持有者产生更新请求并直接发给 CA。CA 处理证书更新请求,如果正确的话,就把新证书发给证书持有者,并将其放入资料库。如果签名是无效的,或者请求实体当前是非法的,或者 CA 认证业务说明或证书策略不允许更新请求,CA 将会拒绝该证书更新请求。如果 CA 拒绝了更新请求,它将会向请求实体报告失败并说明原因。

b) 颁发加密证书

CA 可以支持某个请求者发出的加密证书的证书请求,该请求者拥有该 CA 颁发的有效签名证书。请求者的身份由对请求的数字签名来验证。本标准中规定请求者的密钥对由第三方集中产生并通过带外方式提供给 CA。

在集中密钥管理证书请求中,证书持有者生成一个证书请求,说明自己想要的加密算法,并用

当前得到该 CA 认可的签名密钥来对该请求签名。CA 响应该请求，发还一个证书和加密私钥。

c) 交叉认证

在一定的约束条件下 CA 可向别的 CA 颁发证书。交叉认证的决定是在带外做出的，并且要通过认证业务说明和证书策略检查。每个 CA 都会对它们的使用者的路径验证做出适当的约束。在获得别的 CA 的公钥后，该 CA 生成证书并将其存放到资料库中。

可选的，交叉认证的 CA 之间可以交换证书，构造证书对并把它们放入资料库。

d) 撤销证书

CA 产生和发布证书撤销列表(CRLs)。CRL 中包括所有被撤销但没有到期的证书。可选的，CA 也可以颁发间接的和增量 CRL。被颁发 CRL 的形式由 CA 的认证业务说明决定。

在 CA 为所有的撤销证书仅颁布唯一的 CRL 的情况下：

1) 新的 CRL 产生时，老 CRL 中的全部信息将放到它的里面，那些新被撤销的证书将被加入到新的 CRL。老 CRL 中带 certificateHold 原因代码的证书也将被放入新的 CRL，继续保持同样的原因代码，变成一个不同的原因代码，或是被新 CRL 忽略。如果被忽略就表明 CA 将保证证书主体和公钥之间的绑定。那种决定撤销但还没有被撤销的证书将放入新的 CRL，既使在新 CRL 发布之前它已经到期了。

2) 在这种情况下，CA 只撤销它们自己颁发的证书。(撤销可能来自一个签名请求，或是 CA 自己决定撤销。本标准中不考虑 CA 自己撤销证书的情况)撤销请求的签署者或者是证书持有者，或者是代表证书持有者或证书持有者组织的权威实体(比如被授权的 RA)，在把证书放入 CRL 之前，CA 要确认验证撤销请求。撤销请求的合法性包括请求签名的合法性。对 RA 签名的撤销请求进行带外验证，根据证书策略是可选的。

CA 发布 X.509 版本 2 的 CRL(版本 2 的 CRL 对应版本 3 的证书)。字段和扩展，以及赋给它们的值，要与 6.3.2 一致。产生和签署了 CRL 之后，CA 要把它放到资料库中去。

e) 颁发证书、交叉证书和 CRL

CA 应当能够颁发证书，交叉证书对和 CRL 以供 PKI client 检索。CA 也应能颁发 CA 证书、交叉证书对、CRL 和终端实体的加密证书。终端实体数字签名证书的颁发是可选的。用于更新目录的机制超出本标准范围。

f) 请求 CA 证书

CA 应能向层次更高的 CA 申请证书，以支持 PKI 的层次信任模型。这个请求可以参考 6.6.2 中的描述。这个证书请求将通过 6.2.3.4 中描述的 basicConstraints 扩展来把该请求 CA 确定为一个实体。

5.2.3 电子事务集合

提供证书管理服务所需的电子事务如下：

- 处理终端实体的证书请求和证书撤销请求；
- 将证书和 CRL 放入资料库；
- 为了验证签名的有效性从资料库中检索证书和 CRL。

CA 处理在 CertReq 消息中的 RA 发起的注册请求。CertReq 消息由 RA 在 PKIProtection 结构里签名。通过签署请求消息，RA 保证证书持有者的身份并且确认它拥有相应私钥。CA 以 CertRep 消息回复 RA 或者证书持有者。如果接受请求，CertRep 中包含新的证书。如果拒绝请求，报文中包含错误代码。(见 6.6.2)

CA 支持自我注册证书请求，此时请求实体还不是证书持有者。CA 要求请求实体利用带外方式从 RA 获得的秘密产生认证信息。这一认证信息代替 RA 的签名保证请求者的身份是经过 RA 核实的。以带外方式从 RA 获得的秘密可以用来产生 mac 或者带密钥的散列函数的对称密钥。请求实体产生

CertReq 消息并且用相应的私钥签名。这一消息利用从 RA 获得的秘密加以保护。之后，CA 产生 CertRep 消息，如果请求消息正确，则 CertRep 中包含新的证书。如果拒绝请求，报文中包含错误代码。(见 6.6.3 和 6.6.4)

CA 处理以 CertReq 消息形式出现的证书更新请求。这种消息被请求实体发到 CA。这个消息包括证书持有者的标识名，当前证书的序列号，新的公钥以及拥有相应私钥的证明。消息可以包括预定的有效期和建议的密钥 id。这个消息可用证书持有者的有效证书(未过期，未被撤销)私钥签署。CA 用 CertRep 消息回复申请者。这个消息包含一个新证书或者错误代码。如果颁发新证书的话，证书中将包含证书持有者的标识名和新的公钥。CA 可以自由修改请求中的合法期限。如果申请消息中没有密钥标识符，CA 将产生一个(见 6.6.5)。

CA 应该接收从 RA 发来的 RevReq 消息。RevReq 消息应该包括证书序列号或证书持有者的标识名。CA 用 RevRep 消息回应。这个消息应该包括状态和失败信息，也可以包括撤销证书的附加细节。

CA 支持由第三方集中产生密钥对(并通过带外方式提供给 CA)的加密证书请求。支持对第三方集中产生的密钥对颁发加密证书的 CA 要处理 CertReq 消息形式的请求。这些消息由请求证书的终端实体发送给 CA。这些消息应当包括证书持有者的可辨别名，当前证书的序列号。可选的，该消息可包括一个有效期。该消息用证书持有者的未过期，未被撤销的签名证书的相应私钥签名。之后 CA 以 CertRep 消息的形式响应请求者。该消息包括新证书和加密私钥，或是错误代码。如果颁发证书的话，该证书就包括证书持有者的可辨别名和新的公钥。CA 可以随意修改请求中希望的有效期。

CA 应该把 CA 证书、交叉证书对、CRL 以及它颁发的终端实体的加密证书放入资料库。

5.3 注册机构(RA)

5.3.1 概述

RA 保证请求证书的实体的身份。RA 可以通过要求实体用物理令牌来和 RA 进行物理上的接触或通过带外机制来验证身份。当实体物理上接触 RA 时，RA 也要通过验证一个被签名的消息来验证它们拥有与公钥相应的私钥材料(见 6.6.2)。

RA 应该验证实体拥有一个完整的密钥对。在密钥对和实体身份被验证之后，RA 签署并发送一个电子证书请求给相应的 CA。

没有与 RA 进行过物理接触的证书请求者，在进行证书请求时，必须具有 RA 提供给他的认证信息。这一信息用于实体在自我注册请求中向 CA 证明自己的身份。本标准不对实现自我注册请求的带外事务的格式和内容进行规定。

RA 可以对 CA 授权它们管理的实体证书请求证书撤销。RevReq 的格式在 6.6.7 中定义。RA 功能可以与 CA 在一起也可以在一个不同的设施中执行。

RA 本身既包括证书持有者的功能——请求、撤销和更新由 CA 颁发的证书(它得是该证书的主体)(见 5.4)，又包括客户的功能——检索证书和 CRL 以及验证证书认证路径(见 5.5)。

5.3.2 与互操作性有关的 RA 功能要求

RA 应该执行下列功能：

- 接受和验证证书请求；
- 向 CA 发送证书请求；
- 从资料库检索证书和 CRL；
- 产生证书撤销请求。

RA 应该能够连同 CA 的证书一起把新签署的证书发给证书持有者。

RA 应该可以代表不再拥有它们的私钥并且怀疑该私钥已泄露的证书持有者产生并签署证书撤销请求(丢失而并不认为已泄露的签名密钥不是必须被撤销；这由策略决定。注意丢失的保密性的密钥无论如何都必须被撤销，否则发送方就会加密和传输接受方无法解秘的消息)。如果 CA 的认证业务说明允许，RA 也应该可以代表证书持有者的组织产生并签署证书撤销请求。撤销请求由后来把它们发送

给颁发证书的CA的RA所签署。

5.3.3 事务集合

RA所用电子事务能够实现终端实体证书的请求、发送和撤销，以及为了验证签名而从资料库中检索证书和CRL。下面将给出这些事务的一个概括；它们将在6.6中具体地描述。

RA接收来自潜在证书持有者的CertReq格式的证书请求。CertReq消息被潜在证书持有者在PKIProtection结构中所签署。在审查了请求者的凭证并确认该潜在证书持有者拥有相应的私钥之后，RA抽取公钥信息并且用RA的名字和签名建立一个新的CertReq消息。RA把该消息发送给CA。RA应该可以向证书持有者提供该CA的证书。

RA可以从CA接收CertRep消息。如果证书请求被拒绝，RA将审查从CA发来的错误代码并可能会发送一个新的请求。如果一个证书请求被接受了，RA可以向证书持有者提供新的证书。

RA应该在不再拥有它们自己私钥的证书持有者或证书持有者所在组织的要求下，产生撤销请求。通过签署这个请求，RA保证请求者的身份。RA应该可以产生RevReq消息，包括证书序列号或证书持有者的可辨别名。这个RevReq消息应该被一个RA所签署。CA应该用RevRep消息回应RA。

这个消息应该包括状态和失败信息，并且可以包括关于被撤销证书的附加细节。如果证书被撤销了，RA应该提供这个信息给请求者。如果请求被拒绝了，RA将审查错误代码并且可能再产生该请求。

5.4 证书持有者规范

5.4.1 概述

PKI为证书持有者提供证书管理功能。证书持有者包括CAs、RAs和其他的终端实体。终端实体可能是个人用户和计算机系统(例如路由器和防火墙)，也可能是应用程序(CAs和RAs除外)。

PKI证书持有者生成签名并且支持PKI事务来获取、更新和撤销他们的证书。

5.4.2 与互操作性相关的PKI证书持有者功能要求

证书持有者的功能：

- 生成签名；
- 生成证书请求；
- 请求证书撤销；
- 请求证书更新(可选项)。

证书持有者同时也是客户，因此它也具备5.5中定义的客户功能。

5.4.3 证书持有者事务集合

证书持有者的事务能使证书持有者请求新的证书，以及撤销当前持有的证书。所有的证书持有者事务都由发放证书的CA及其CA授权的RA执行。

证书持有者应该能够请求撤销他们自己的证书。这个事务由CA执行，允许证书持有者签署他们的证书撤销请求。证书持有者为每个他们希望撤销的证书产生一个RevReq消息并发送给CA。RevReq消息要包括撤销原因。CA产生RevRep消息并返还给证书持有者。这个事务过程在6.6.7中有详细描述。

可选的，证书持有者能够实现证书更新请求。这个事务由CA执行，允许证书持有者签署自己的证书请求(不需要RA验证其身份)。CA可以支持该项事务，但是它的具体使用由认证业务说明决定。在不与RA交互的情况下请求一个新的证书，证书持有者产生一个CertReq消息，并且使用新的和当前的私钥进行签名。CA产生CertRep消息，如果请求成功，就包含一个新的证书。如果请求被拒绝，就包含错误代码。

证书持有者能够实现自我注册的证书请求。

证书持有者也可以实现加密证书请求。在此事务中，证书持有者产生一个证书请求，说明希望的密钥管理算法，并用一个有效的签名密钥对该请求签名。CA响应一个证书和加了密的私钥或是错误代码。

5.5 客户规范

5.5.1 客户概述

PKI客户使用PKI来为证书持有者和证书使用者，包括CA和其他的终端实体提供证书处理功能。终端实体也可以包括RA，个人用户和计算机系统(如路由器和防火墙)。

在最小意义上，PKI客户验证签名，获得证书和CRLs，并验证证书认证路径。同时具有证书持有者身份的客户也能产生签名，也可以支持撤销或更新证书的PKI事务。

5.5.2 与互操作性相关的PKI客户功能要求

客户的最小功能集合：

- 验证签名；
- 从查询服务器中检索证书和CRLs；
- 验证证书认证路径。

5.5.3 PKI客户事务集合

客户事务使客户能从资料库中获得证书和CRLs。所有的事务都与证书资料库有关。所有的客户必须支持以下事务：

- 检索证书——这个事务允许一个用户利用LDAP绑定目录服务或指定资料库，并根据主体 名或证书序列号和颁发者的名字检索证书；
- 检索CRL——这个事务允许一个用户利用LDAP绑定目录服务或指定资料库，来检索一个特定CA的当前CRL或是某个特定的CRL。所有的客户都必须支持轻型目录访问协议(LDAP)，以检索证书和CRLs。事务的详细规定参见RFC1777。

6 数据格式

6.1 数据格式概述

必须为PKI组件之间的互操作定义基本的数据格式。数据格式包括证书、CRL和事务格式。本标准包括了PKI组件之间、PKI client和PKI组件之间所有事务的数据格式。

6.2 证书格式

本标准使用GB/T 16264.8—2005的证书格式。虽然ITU-T建议的X.509的修订版没有公布，但版本3的格式已经被广泛采用，并且ANSI的X9.55和IETF的RFC 2459都对其作了详细说明。GB/T 16264.8—2005的证书包括以下内容：

- Version 版本；
- Serial Number 证书序列号；
- Issuer Signature Algorithm 颁发者签名算法；
- Issuer Distinguished Name 颁发者可辨别名；
- Validity Period 证书有效时间；
- Subject Distinguished Name 主体可辨别名；
- Subject Public Key Information 主体公钥信息；
- Issuer Unique Identifier 颁发者唯一标识符(可选，本标准不使用)；
- Subject Unique Identifier 主体唯一标识符(可选，本标准不使用)；
- Extensions 证书扩展(可选)；
- Issuer's Signature on all the above fields 颁发者对以上所有域的签名。

6.2.1 证书字段

X.509证书语法的ASN.1定义见附录A。出于签名考虑，证书用ASN.1 DER编码表示。ASN.1 DER是一个为每个元素赋予标签、长度和值的编码系统。详见ISO/IEC 8825.1—2002。

以下介绍GB/T 16264.8—2005的证书。除了可选的subjectUniqueID和issuerUniqueID字段，

CA 应该产生下面这些字段，而且客户根据 GB/T 16264.8—2005 可以处理它们。CA 可以不颁发包括 issuerUniqueID 和 subjectUniqueID 的证书。客户也可以不处理 subjectUniqueID 和 issuerUniqueID。他们可以拒绝那些含有这两部分字段的证书。

a) Version

version 字段表示证书的版本号。该字段的值应为 2，以表示版本 3 证书。

b) Serial Number

serialNumber 是 CA 给每个证书分配的一个整数。对一个给定的 CA，它提供的证书序列号应该是唯一的(即，颁发者和证书序列号唯一标识一个证书)。

c) Signature

signature 字段包含了一个算法标识符，用于表示签署证书的算法。此字段包括一个 algorithmIdentifier，用于传递参数。在 6.2.2.2 中列出了 algorithmIdentifier 的内容。证书不应该使用 signature 字段去传递参数(看下面的Subject Public Key Information)，因为该字段没有被颁发者的签名保护。

d) Issuer Name

issuer Name 字段提供了签署证书机构的全球唯一标识符。颁发者的名字是一个 X.500 标识名。该标识名由属性类型和属性值组成。通常属性类型由 X.500 系列建议定义，属性值为 DirectoryString 类型。

DirectoryString 可选择 PrintableString，TeletexString，BMPString，UniversalString 和 Utf8String 中的一种。PrintableString 是一个基本拉丁字符集，可使用大小写字母，数字和少量特殊字符。TeletexString 是 PrintableString 的扩展集，在拉丁字符中加入了重音符和日本字符。BMPString 是双字节字符集，它满足绝大多数的欧洲字符集。UniversalString 是多字节字符集，包括了所有的主要字符集。Utf8String 是 UniversalString 的替代编码。

当创建新名字时，CA 要从 PrintableString，BMPString，和 Utf8String 里选择最严格的类型来创建 DirectoryString。也就是说，仅需基本拉丁字符的 AttributeValue 一般就使用 PrintableString。包含重音符拉丁字符的 AttributeValue 就使用 BMPString。当 BMPString 字符集不够用时才使用 Utf8String。

仍然保留 TeletexString，和 UniversalString 是为了向后兼容性的需要，不应该在新主体的证书中使用它们。但是，此种类型可能会在已经颁发的证书中使用。Clients 可能会收到使用这些类型的证书。

可选的名字可放在 issuerAltName 扩展字段，一些 GB/T 16264.8—2005 证书的使用者希望 issuer 字段为空。然而，遵从本标准的证书中的这个字段都含有 X.500 的可辨别名。

e) Validity

validity 字段说明了证书开始生效的日期(notBefore)和变成无效的日期(notAfter)。validity 字段的时间表示形式用 UTCTime 或 GeneralizedTime。

对于 1950 年到 2049 年(含)的日期，validity 字段一般使用 UTCTTime。UTCTTime 使用格林威治时间，而且要精确到秒。秒要清楚表示，即使为零。UCTCTime 的格式为 YYMMDDHHMMSSZ。

年字段做如下规定：

1) 如果 YY 等于或大于 50，年代为 19YY；

2) 如果 YY 小于 50，年代为 20YY。

对于 2050 年以后的日期，validity 字段一般用 GeneralizedTime。GeneralizedTime 使用格林威治时间，而且要精确到秒。秒要清楚表示，即使为零。GeneralizedTime 的格式为 YYMMDDHHMMSSZ。GeneralizedTime 不能包括分秒。(1950 年前的日期对本标准来说无效，所以

不予考虑。)

f) Subject Name

subject Name 字段的目的是给证书主体提供唯一的标识符。主体名使用 X.500 可辨别名。像在 issuer name 中描述的一样,在构建 DirectoryString 时 CA 使用最严格的选择。可替代的名字可放在 subjectAltName 扩展中,GB/T 16264.8—2005 证书的使用者希望本字段为空。然而,遵从本标准的证书都在这个字段里带有主体的 X.500 可辨别名。

g) Subject Public Key Information

SubjectPublicKeyInfo 字段是用来携带公钥和公钥使用的算法的。它包括 subjectPublicKey 字段和 algorithmIdentifier 字段,algorithmIdentifier 字段有 algorithm and parameters 次级字段。

h) Unique Identifiers

证书里的 subjectUniqueIdentifier 和 issuerUniqueIdentifier 字段是用来处理主体名和颁发者名被过时重用的可能性。CA 不会颁发含有这些唯一标识符的证书。PKI 客户也不会被要求去处理含有这些标识符的证书。当然,如果它们不处理这些字段,他们会拒绝包含这些字段的证书。

i) Extesion

extension 字段的增加是 GB/T 16264.8—2005 证书最主要的改变。扩展有三部分:extnId,标识该扩展,critical,说明该扩展是 critical 或 noncritical 的,extnValue,扩展值。一个证书可以包括任意多个扩展字段,其中也包括局部定义的扩展。如果设置了 critical 标志,就要求客户或者能够处理该扩展字段,或者不能验证该证书。

在 GB/T 16264.8—2005 中有完整的扩展标准。6.2.3 详述了这些标准扩展在本标准中的使用。

j) Issuer's Signature

实际的证书签名使用 SIGNED 参数类型,扩展为被签名的数据的 SEQUENCE(例如,证书),算法标识符,以及一个 BIT STRING 形式的实际签名。algorithmIdentifier 标识了用来对证书签名的算法。尽管这个 algorithmIdentifier 字段包括了一个 parameters 字段,在理论上可以用来传递签名算法的参数(见 6.2.2.3),但它本身并不是一个被签名的对象。parameters 字段不能用来传递参数。当需要获得验证签名的参数时,应当从颁发证书的 CA 的证书的 subjectPublicKeyInfo 字段中获得相应参数。

6.2.2 加密算法

6.2.2.1 加密算法概述

本标准使用四类算法:散列函数、数字签名算法、消息认证算法和对称加密算法。当描述数字签名算法时,通常与散列函数一起介绍。当描述证书中的公钥时,散列函数被忽略。这就允许当散列函数被另一个更强的算法替换时,证书也能使用。

要求一个 PKI 组件至少应该实现其中一个数字签名算法。对于简单的 PKI 客户组件来说,能验证由其中一个算法生成的签名已经足够了;要求其他的组件能够产生和验证由其中一个算法生成的签名。遵守本标准的 PKI 组件应该支持一个加密算法。

允许相应的组件使用额外的算法,甚至当它们不能使用所有的这些算法时。比如,支持这里的一个密钥协定算法的客户可以使用别的密钥传递算法,即使它不支持这里的密钥传递算法。

6.2.2.2 散列函数

散列函数用于为证书和 CRL 产生数字签名,它也用于将共享秘密散列为报文确认码。

本标准中建议使用一种散列函数。以 SHA-1 为例,它是通过对象标识符以及数字签名算法表示的。SHA-1 的 ASN.1 对象标识符是:

sha1 OBJECT IDENTIFIER ::= {
iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }

只要该对象标识符在算法类型中出现，相应的参数域应该被忽略，如果出现，必须为 NULL。

6.2.2.3 数字签名算法

GB/T 16264.8—2005 中指定了发放证书的签名算法和其他主体的加密算法。这两种算法可以是不同的。本标准以 RFC 2313 中的 RSA 算法为例。

RSA 的签名算法在 RFC 2313 中定义。RSA 也可使用几种散列算法。本标准中引用的 RSA 定义为：

```
pkcs-1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) US(840)
                    rsadsi(113549) pkcs(1) 1 }
sha-1WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1(1) 5 }
```

在 RSA 签署的证书和证书撤销列表中，它应出现在 SIGNED 类型中和 signature 字段中。只要实体的标识符作为算法标识符出现，那么参数的值必须是 NULL。

当证书和证书撤销列表签署时，签名将按下列规则生成和加密：证书和证书撤销列表按 ASN.1 DER 加密，并作为 SHA-1 的散列函数的输入。SHA-1 散列函数的输出为 OCTET STRING 并按照 RSA 的加密算法加密。签署证书时，RSA 生成一个整数 y。y 的值按照 ASN.1 的 BIT STRING 输出。在 y 的 signature 字段包含了证书或证书撤销列表。（通常分为两步，整数 y 先转化为 octet string。然后再转化为 bit string。）

当 CA 发放了一个证书，并且它的 subjectPublicKeyInfo 字段包含了 RSA 的公钥，那么对象标识符 rsaEncryption 将作为 algorithmIdentifier 出现在 subjectPublickeyInfo 字段中，来标识 RSA 的公钥。

```
rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1}
```

只要 rsaEncryption 对象标识符在算法类型中出现，那么参数字段必为 NULL。

RSA 的公钥必须为 ASN.1 的 RSAPublicKey:的类型。

```
RSAPublicKey ::= SEQUENCE {
modulus                    INTEGER, -- n
publicExponent        INTEGER -- e}
```

系数为 n，publicExponent 为公共指数 e，RSAPublicKey 编码后为 BIT STRING subjectPublicKey 的值。

RSAPublicKey 既出现在 RSA 的签名密钥中，也在 RSA 的加密密钥中出现。虽然不禁止只使用一个密钥来进行加密和签名，但不提倡这样做。

6.2.2.4 消息认证算法

本标准建议支持两种消息认证算法。以 DES-MAC 和 SHA1-MAC 为例，推荐使用 SHA1-MAC，DES-MAC 是为了保证向后的兼容性。

SHA1-MAC

本算法通过为数据计算一个 SHA-1 HMAC 值（在 RFC2104 中详细定义）而提供完整性保护。它的算法标识符如下：

SHA1-HMAC OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 8 12 }

本标准中使用的 mac 长度为 96bit。

DES-MAC

本算法通过为数据计算一个 DES MAC 值（在 FIPS-113 中详细定义）而提供完整性保护。它的算法标识符如下：

DES-MAC OBJECT IDENTIFIER ::= {
iso(1) identified-organization(3) oiw(14) secsig(3) algorithm(2) 10

-- MAC 的长度是一个整数型的参数，在本标准中限制为 32 }

参数域包含一个整数，表明了 MAC 的长度，在本标准中规定该值取为 32。

6.2.2.5 对称加密算法

本标准使用了对称密钥算法来实现挑战－响应协议和保护私钥传输。在本标准中，以使用 tDEA (Triple Data Encryption Algorithm)算法的 ECB 模式为例说明对称密钥算法的使用。

tDEA 算法在 ANSI X9.52 中定义。tDEA 算法基于 DES 算法，使用 3 个 56 bit 的密钥：K1、K2 和 K3。在本标准中，使用双密钥模式，K1＝K3。tDEA 算法的密钥绝对不能出现在 GB/T 16264.8—2005 证书中。有的情况下，会在 PKIMessage 中传输 tDEA 的密钥，这时候，tDEA 的密钥必须是经过加密的。

下文说明了 tDEA 的双密钥、ECB 模式的加密方法和密钥的编码格式。

tDEA 算法有 AlgorithmIdentifier，而且要有 tECB 的 OID。双密钥选项在 ECBParams 结构中指明。

ASN.1 表示如下：

```
TDEAIdentifier ::= AlgorithmIdentifier {{ TDEAModes }}
TDEAModes ALGORITHM-ID ::= {
{ OID tECB PARMS ECBParms } | -- mode 1 --
{ OID tCBC PARMS TDEAParms } | -- mode 2 --
{ OID tCBC-I PARMS TDEAParms } | -- mode 3 --
{ OID tCFB PARMS CFBParms } | -- mode 4 --
{ OID tCFB-P PARMS CFBParms } | -- mode 5 --
{ OID tOFB PARMS TDEAParms } | -- mode 6 --
{ OID tOFB-I PARMS TDEAParms }, -- mode 7 --
...
}
ECBParms ::= TDEAParms (WITH COMPONENTS {
..., ivGeneration ABSENT })
TDEAParms ::= SEQUENCE {
keyingOptions KeyingOptions OPTIONAL,
ivGeneration [0] IVGeneration OPTIONAL
}
-- 本标准中只支持双密钥
KeyingOptions ::= BIT STRING {
option-1 (0), -- (3-key) K1, K2 and K3 are independent keys
option-2 (1), -- (2-key) K1 and K2 are independent and K3 = K1
option-3 (2) -- (1-key) K1 = K2 = K3
}
id-ansi-x952 OBJECT IDENTIFIER ::= {
iso(1) member-body(2) us(840) ansi-x952(10047) }
Second CRADA Draft, Version 2
mode OBJECT IDENTIFIER ::= { id-ansi-x952 1 }
tECB OBJECT IDENTIFIER ::= { mode 1 }
```

有的情况下，需要对 tDEA 的密钥进行编码，就是两个密钥的直接串接。TwoKeys = [K1 | K2]，TwoKeys 的最高位比特是 K1 的最高位比特。

6.2.3 证书扩展

6.2.3.1 证书扩展概述

在 GB/T 16264.8—2005 中定义了一套标准扩展集合。扩展由三部分组成：扩展名称、关键性标识位和扩展取值。在 GB/T 16264.8—2005 中做了如下规定：客户如果不能处理设置了关键性标识位的扩展，就不能验证证书是否有效。

标准化的扩展可以分为四类：密钥和策略信息；主体和颁发者的特征；验证路径限制；CRL 标识扩展。

6.2.3.2 密钥和策略信息

此类扩展用于区分特定的公钥和证书，它们可以用于区别具有多个证书的证书认证机构（CA）中某一特定的公钥/证书，有助于客户查找某个特定的 CA 证书，以便建立验证路径。这些扩展可以限制密钥用途，提供 CA 证书中关于策略映射的信息。

a） 权威密钥标识符

扩展 authorityKeyIdentifier 提供了一种区别签名证书的特定私钥的手段，标识性信息既可以基于密钥标识符，也可以基于颁布者名字和序列号。在本标准中，使用密钥标识符的方法。本扩展用于具有多个签名密钥的颁发者（既可能是多个密钥对，也可能是正在进行密钥更换），证书认证机构（CA）应当能够产生本扩展，而且客户应能查找和验证具有多个数字签名密钥的证书颁发者 CA 的验证路径。建议客户既能够处理密钥标识符，又能够处理由证书颁发者和证书序列号组成的密钥标识符，便于找到验证路径。

b） 主体密钥标识符

本字段用于区别一个主体所拥有的多个密钥，在每个已颁发的证书中都应当包含本字段，本扩展应当是非关键的。

c） 密钥用途

扩展 keyUsage 定义了证书中密钥在使用上的限制，这种限制是基于策略和/或用途的（如签名，加密）。CA 应当能产生该扩展，而客户应当能处理该扩展。如果 keyUasge 定义为 BIT STRING，所有遵从本标准的 CA 都应当在终端实体证书中设置一个值，例如，一个终端实体证书中的 KeyUsage 不应当既是 digitalSignature 又是 dataEncipherment，本扩展应当设置为关键性的扩展。

d） 私钥使用期限

扩展 privateKeyUsagePeriod 仅适用于数字签名密钥。在私钥使用期限之外的签名是无效的，CA 可以产生此扩展，但客户不要求处理此扩展。

e） 扩展的密钥使用范围

扩展 extendedKeyUsage 用于定义在应用中对证书密钥的使用限制，如果使用此扩展，就不考虑互操作的因素。符合本标准的 PKI 组件不要求支持该扩展。

f） 证书策略

扩展 certificatePolicies 包括一个或多个对象标识符（OIDs），每个 OID 代表颁发本证书的一个策略。CA 应当能够产生带有一个或多个证书策略 OID 的证书，CA 应当支持一个特殊策略 OID，即任意策略 anyPolicy。anyPolicy 的 OID 可以认为是一个通用符，能够匹配任何策略。客户应当能够处理 policyIdentifier 字段，并与可接受的策略列表（策略列表取决于相应的应用程序）中的标识进行比较。如果提供了一个可接受策略列表，并且至少要有一个列表中的策略或其映射策略存在于 certificatePolicy 中时，客户才能验证证书认证路径。如果在证书策略中存在特定策略标识符 anyPolicy，并且没有禁止任何策略（见 6.2.3.4），就可以认为列表中的所有策略都可以接受。

要求符合本标准的组件能处理 certificatePolicy 的子域 policyQualifiers，并且支持 id-pkix-cps

和 id-pkix-unotice(见 RFC2459)。符合本标准的 CA 不要求能产生该子域。

g) 策略映射

此非关键性扩展用于 CA 证书。它罗列了对象标识符对;每对标识符包括 issuerDomainPolicy 和 subjectDomainPolicy。它意味着颁发 CA 认为其 issuerDomainPolicy 等价于主体 CA 的 subjectDomainPolicy。CA 应当能产生扩展 policyMappings,客户也应能够处理该扩展。

6.2.3.3 证书主体和颁发者特征

扩展 subjectAltName,issuerAltName, subjectDirectoryAttributes 和 authorityInformationAccess 都是非关键性的。它们提供了关于主体和颁发者的其他名称和特征的附加信息。

a) 别名

subjetAltName 和 issuerAltName 扩展提供了证书主体和证书颁发者的附加信息,这些附加信息包括 RFC822 名称(电子邮件地址)、DNS 名称和统一资源标识符(URI)。扩展中可以包括多个名称。如果需要在证书中提供这些信息,就应当使用扩展 subjectAltName 和 issuerAltName。

根据本标准,扩展 subjectAltName 和 issuerAltName 应当是非关键的。识别这些扩展的应用没有必要能支持所有可能的选择。如果应用不能识别证书中的别名,就可以忽略该扩展。

在本标准中,扩展 issuerAltName 中包括 URI 信息,URI 明确了颁发者证书的位置,这可以用来验证数字证书中的签名。在本标准中没有规定 subjectAltName 中包含的信息的意义。

如果证书认证机构(CA)的证书不能从 X.500 目录服务中得到,CA 就应当在证书中的别名扩展中包含相应的 URI 信息,以指出签名者证书的位置。要求客户能处理 URI 别名格式,必须能够识别 LDAP URL[RFC1959]。不要求客户能识别其他 URI 格式。

b) 主体目录属性

扩展 subjectDirectoryAttributes 可以包含任何关于主体的 X.500 目录属性信息。本扩展是非关键性的。本扩展的实现和使用是可选的。

c) 权威机构信息存取

扩展 authorityInformationAccess 中保存如何获得颁发 CA 和服务的信息。信息和服务可以包括在线验证服务和 CA 策略数据。(CRL 的位置信息不在本扩展域显示,该信息由 cRLDistributionPoints 扩展提供。)

6.2.3.4 验证路径限制

扩展 basicConstraints, nameConstraints 和 policyConstraints 用于限制有效的验证路径。

a) 基本约束

扩展 basicConstraints 通过 CA 部分的信息指定证书的实体是否是 CA,通过 pathLenConstraint 部分的信息指定验证路径的长度。CA 应当在证书中能产生扩展 basicConstraints,客户也应能够处理本扩展。只有在 CA 为 True 的时候,pathLenConstraint 才有意义。

在所有 CA 证书中都应当包括 basicConstraints 扩展,而且 CA 应当设置为 True。在终端实体的证书中也可以包括扩展 basicConstraints。如果在终端实体的证书中出现扩展 basicConstraints,它应当是一个空的序列值。在所有 CA 证书中的扩展 basicConstraints 应当设置成关键性的。

b) 名字约束

扩展 nameConstraints 只能用在 CA 证书中,它指定证书的验证路径中所有后续证书的名字范围。CA 应当在证书中能产生扩展 nameConstraints,客户也应能够处理该扩展。该扩展应设置为关键性的。

c) 策略约束

扩展 policyConstraints 只用于 CA 证书。它可以以两种方式对策略解释进行限制:既可以限

制策略映射，又可以要求在验证路径中的每个证书包括一个可接受的策略标识符。

本扩展包括两个字段：inhibitPolicyMapping 和 requireExplicitpolicy。如果 inhibitPolicyMapping 存在，它的值代表验证路径中的策略映射不再有效之前的其他证书的个数。如果 requireExplicitPolicy 存在，在后续的证书中应当明确包含可接受的策略标识符。RequireExplicitpolicy 的值代表在验证路径中要求明确的策略前的其他证书的个数。

CA 应当在颁发证书中包括本扩展，客户应能够处理本扩展。该扩展应设置为关键性的。

d) 禁止 anyPolicy

本扩展禁止在今后的证书中使用 anyPolicy。此关键性的扩展可以出现在颁发给 CA 的证书中。禁止策略意味着特殊策略标识符 anyPolicy OID(值为{2 5 29 32 0})不能和其他证书策略匹配。该扩展的值是 INTEGER，代表在验证路径中不再允许 anyPolicy 之前的其他证书的个数。例如，1 意味着 any-policy 只能在本证书的主体所颁发的证书中处理，在验证路径中的其他证书中不能使用。CA 应当能够产生此扩展，客户应能够处理本扩展。

6.2.3.5 CRL 标识扩展

本类扩展包含如何获得证书撤销列表(CRL)的信息。通过标识 CRL 和颁发 CRL 者的名称(颁发者可能不是产生证书的 CA)的方式，可以将大的 CRL 列表划分为多个小 CRL 列表。

CRL 分发点：

CRLDistributionPoints 扩展标识 CRL 分发点，或指导客户如何验证一个已撤销的证书。本扩展由三部分字段组成：distributionPoint，reasons 和 cRLIssuer。

a) DistributionPoint 存放如何获得 CRL 的位置信息。如果本字段缺省，CRL 分发点的名字就是颁发者名字。在 CA 颁发的 CRL 很大的情况下，此字段提供了一种将 CRL 划分为多个可管理片断的机制。

b) Reasons 存放相应的 distributionPoint 分发的 CRL 撤销的原因。如果 reasons 不出现，在 distributePoints 分发的 CRL 中可以包括由于任何原因而撤销的证书。不要求客户能处理 reasons。

c) CRLIssuer 确定分发并签名 CRL 的权威机构。如果该字段不出现，CRL 颁发者与证书颁发者相同。使用本字段可以建立统一的 CRL，包括由多个 CA 颁发的证书。

CAs 应当能够产生带有 distributePoint 部分的 cRLDistributionPoints 扩展。如果不能从公共的 X.500 目录服务中获得 CRL 列表，CA 应当在 distributionPoint 中用 URI 别名格式来标注相应 CRL 列表的位置。要求客户能够处理 cRLDistributionPoint 扩展。必须能够识别 URI 格式，至少能够处理 LDAP URI 格式。如果使用了 cRLIssuer，要求客户能够使用分发点并验证 CRL。在 6.3.3 中对分发点做进一步的讨论。

6.3 证书撤销列表

6.3.1 证书撤销列表概述

证书撤销列表(CRL)用来列出那些已被撤销或冻结的但并未过期的证书。证书撤销的原因多种多样，例如日常的管理撤销(当证书的主体离开了发放组织，或责任和证书属性发生了变化)，或私钥被泄密。“冻结”是指 CA 不再确保证书主体和公钥之间的绑定。

X.509 v2 证书撤销列表格式增加了一些可选扩展，在概念上与证书扩展相似。CAs 应能产生如下的 X.509 v2 CRLs，当验证证书认证路径时客户也应能处理它们。颁发 CRLs 的 CA 不一定必须是颁发该撤销证书的 CA。一些 CAs 只负责发放 CRLs。X.509 v2 CRL 包含以下信息：

- Version 版本；
- Issuer Signature Algorithm 颁发者签名算法；
- Issuer Distinguished Name 颁发者可辨别名；
- This Update 本次更新；

- Next Update　下次更新；
- Revoked Certificates　撤销的证书，零个或多个以下序列的序列：
  - ◆ Certificate Serial Number　证书序列号，
  - ◆ Revocation Date　撤销日期，
  - ◆ CRL Entry Extensions　CRL Entry 扩展(可选)；
- CRL Extensions　CRL 扩展(可选)；
- 颁发者对以上字段的签名。

6.3.2 CRL 字段

X.509 v2 CRL 的 ASN.1 句法见附录 A。对于签名计算，签名的输入数据是 ASN.1 DER 形式的编码。ASN.1 DER 编码对每一个元素来说都是标签、长度、值的编码系统。以下各项描述了 X.509 v2 CRL 的使用：

a) Version

这个字段描述了编码后的 CRL 的版本。该字段的值应是 1，表明是 v2 CRL 。

b) Signature

该字段包含签署 CRL 的算法的算法标识符。它的内容与证书的 signature 字段一样。关于这个字段的信息见 6.2.1 中关于 signature 字段的定义。CRL 可以被 6.2.2.2 中标识的任何算法签名；通常，CA 使用同一算法来对证书和 CRL 签名。

c) Issuer Name

issuer 字段提供了签署 CRL 的 CA 的全球唯一标识名。颁发者的名字是 X.500 可辨别名。本标准不支持 CRL 颁发者名字为空的 CRL。

d) This Update

thisUpdate 字段指定了该 CRL 的日期。此字段可以是 UTCTime 或 GeneralizedTime。对于本标准，thisUpdate 字段遵从证书的 validity 字段的规则。(见 6.2.1)

e) Next Update

nextUpdate 字段指定了下一个 CRL 颁发的日期。下一个 CRL 颁发的日期可以在指定日期之前，但不可在指定日期之后。此字段可以是 UTCTime 或 GeneralizedTime。对于本标准，nextUpdate 字段遵从证书的 validity 字段的规则。(见 6.2.1)

f) Revoked Certificates

revokedCertificates 字段是一个已经被撤销的证书的列表。每个被撤销证书包含：

1) 证书序列号(在 userCertificate 字段中指出)。它包含被撤销证书的 serialNumber 字段的值。它必须与颁发 CA 的名字一起使用以识别一个已经被撤销但还未到期的证书。
2) 包含撤销日期的 revocationDate 字段。本字段取值遵从证书的 validity 字段的规则。(见 6.2.1)
3) CRL Entry 扩展(可选)(见 6.3.4)。它可以给出证书撤销的原因，说明证书无效日期，也可以说明发放该被撤销证书的 CA 的名字(或许与颁发 CRL 的 CA 不是同一个)。注意：我们假定颁发 CRL 的 CA 和颁发被撤销证书的 CA 是同一个，除非 CRL Entry 扩展包含 certificateIssuer 字段。

6.3.3 CRL 扩展

ISO/ITU 所定义的 X.509 v2 CRLs 扩展提供了对全部 CRLs 附加其他信息的方法。每个 CRL 扩展被设计为 critical 或 noncritical。(关键的或非关键的)如果客户碰到一个不能处理的关键性扩展，将无法对该 CRL 进行验证。

本条描述了应被支持的 CRL 扩展。当 CA 能够产生一个 CRL 的扩展且客户能够处理该扩展时，此 CRL 扩展才有效。

a) 机构密钥标识符

authorityKeyIdentifier，非关键性 CRL 扩展，标识了 CA 用来签署 CRL 的密钥。当 CA 使用多个密钥时该扩展是有用的；它使得不同的密钥得以区分（例如，密钥更新时）。身份的鉴定可基于密钥标识符或发放者名字和序列号。所有的 CRLs 都应有密钥标识名。当发放者拥有多个签名密钥时（或多个密钥对，或是在密钥更新期间），该扩展是很有用的。当发放 CA 有多个签名密钥对时，在所有的 CRL 扩展里都应包括该扩展，并且客户也应能够找到并验证 CRL 验证路径。客户如果要查找证书认证路径，必须能够处理 authorityKeyIdentifier 的密钥标识符或证书发放者名加上序列号。

b) 颁发者别名

issuerAltName ，非关键性 CRL 扩展字段，包含一个或多个 CA 别名。别名一旦出现就放置在 issuesrAltName 里。并非所有的别名格式都要被识别处理，识别不出的别名格式可以被忽略。CA 可在 CRLs 中产生该扩展，然而客户可以不予处理。

c) CRL 编号

cRLNumber，非关键性扩展字段，是一个单调增加的序列号，该 CRL 由 CA 通过特定的 CA 目录入口或 CRL 分布点发放。该扩展可以用来通知证书使用者整个 CRL 的非固定时间发布，或是便于确定何时某一 CRL 替代了另一 CRL。在 CRL 中应该包括该扩展。

d) 颁发分布点

issuingDistributionPoint 字段是非关键性的扩展，决定了一个特定 CRL 的 CRL 分布点。一个分布点就是一个用来检索 CRL 的目录入口，它可以与 CA 的目录入口不同。CA 要用密钥对 CRL 加密。CRL 分布点没有自己的密钥对。

另外，issuingDistributionPoint 字段指定的 CRLs 可能只针对终端实体的证书，或者只针对 CA 证书，或者只针对由于某种特定原因撤销的证书。最后，该扩展也可以确定一个间接 CRL，间接 CRL 是由与发放被撤销证书 CA 不同的 CA 发放的。间接 CRL 包含以下组件：

- distributionPoint，给出了分布点的名字，遵循 X.500 标识名规则；
- onlyContainsUserCerts，布尔变量，表明该 CRL 只包含终端实体证书；
- onlyContainsCACerts，布尔变量，表明该 CRL 只包含 CA 证书；
- onlySomeReasons，一个 ReasonFlag 位串，标明 CRL 中所列证书的撤销原因；如下：
  - keyCompromise，表明密钥泄漏或怀疑密钥泄漏，
  - cACompromise，表明该证书撤销的原因是 CA 密钥泄漏，它只用于撤销 CA 证书，
  - affiliationChanged，表明该证书撤销的原因是证书主体的从属关系改变了，
  - superseded，表明该证书已被代替，
  - certificateHold，表明证书处于冻结状态，可能会被撤销，
  - cessationOfOperation，表明该证书的目的已经不再被需要，但并不是密钥被泄漏；
- IndirectCRL，布尔变量，表明这是一个间接 CRL。

客户应能处理这个字段。

e) 增量 CRL 指示符

deltaCRLIndicator 是一个关键 CRL 扩展，它确定一个增量 CRL。对于那些采用非 CRL 结构存储撤销信息的用户程序来说，增量 CRL 可以有效地节省处理时间。它允许把变化的信息增加到本地数据库里，忽略掉那些本地数据库中已经存在的未改变的信息部分。

BaseCRLNumber 的值表明基础 CRL 的 CRL 序列号，基础 CRL 是生成增量 CRL 的开始点。增量 CRL 包含基础 CRL 和当前 CRL 之间的变化部分。是否提供增量 CRL 要由 CA 来决定。

客户可以利用本地 CRL 和增量 CRL 合成一个 CRL，要注意的是如果本地 CRL 的 CRL 序列

号小于增量 CRL 中的 BaseCRLNumber，合成的 CRL 就不正确。如果增量 CRL 含有一个 CRL 序号扩展，那么合成的 CRL 的 CRL 序号就是该扩展的值。客户和 CA 是否支持增量 CRL 是可选的。

f) CRL 扩展使用总结

表 1 总结了标准 CRL 扩展，表 2 总结了这些标准 CRL 扩展的使用。

表 1 CRL 扩展概要

扩展	使用	Critical
AuthorityKeyIdentifier	标识签署 CRL 的 CA 密钥	No
KeyIdentifier	唯一的密钥标识符；等价于 certIssuer 和 authorityCertSerialNumber	
authorityCertSerialNumber	和 certIssuer 一起使用；两者必须唯一	
IssuerAltName	CRL 颁发者的别名	No*
CRLNumber	CRL 的序列号	No
IssuingDistributionPoint	CRL 分布点的名字；也给出了撤销原因	Yes
DeltaCRLIndicator	标明增量 CRL 并给出序列号	Yes
注：* 表示在 X.509 v2 CRL 中此扩展可以是关键扩展也可以是非关键扩展，为了实现互操作，本标准将该扩展定义为非关键扩展。		

表 2 CRL 扩展在本标准中的使用

扩展	CRL	客户
AuthorityKeyIdentifier	可选	可选
KeyIdentifier	所有的 CRL 中都包括该标识符	可选—用来帮助找到正确的 CA 证书以验证 CRL(1)
CertIssuer	不产生	可选—颁发者/序列号用来帮助找到正确的证书以验证 CRL(1)
certSerialNumber	不产生	
CRLNumber	支持；在所有的 CRL 中	可选
IssuingDistributionPoint	支持	支持
DeltaCRLIndicator	支持	可选
IssuerAltName	支持	可选
注：对 CRL 来说，“支持”意味着 CA 能够颁发含有该扩展的 CRL。对客户来说，“支持”意味着客户能够处理含有该扩展的 CRL。无论客户使用该扩展与否，无论 CRL 含有该扩展与否，当 CA 有多个证书时，客户都应当能够找到签署 CRL 的私钥的相应证书。		

6.3.4 CRL Entry 扩展

X.509 v2 CRLs 所定义的 Entry 扩展提供了获取 CRL 每条附加信息的方法。每个 CRL Entry 扩展被设计为关键性或非关键性。如果不能处理关键性扩展，该 CRL 的验证将是无效的。如果是不可识别的非关键性 CRL Entry 扩展，则可以忽略。

a) 原因代码

reasonCode 是非关键性 CRL Entry 扩展，标明了证书撤销原因。CA 应当能够生成该扩展，但客户是否要处理 reasonCode 扩展则是可选的。下面列举的是 reasonCode 的取值：

- unspecified，未使用；
- keyCompromise，表明密钥泄漏或怀疑泄漏；
- cACompromise，表明该证书撤销的原因是 CA 密钥泄漏，它只用于撤销 CA 证书；

- affiliationChanged，表明该证书撤销的原因是证书主体的从属关系改变了；
- superseded，表明该证书已被新的证书代替；
- cessationOfOperation，表明该证书的功能已经不需要，并不是密钥泄漏；
- certificateHold，表明证书当前不能使用，如果证书的 CRL 的 reasonCode 字段是 certificateHold，那么客户就无法验证证书认证路径；
- removeFromCRL，只与增量 CRL 一块使用，表明应该删除一个存在的 CRLEntry。

b) 过期日期

expirationDate，非关键性 CRL Entry 扩展，表明 Entry 的过期时间。该扩展不在 CRLs 中使用，也不被客户使用。

c) 指令码

instructionCode 是非关键性 CRL Entry 扩展，提供了一个注册过的指令标识符，指令标识符来指出当遇到一个被冻结的证书时要采取何种操作。本扩展不在 CRL 中使用。

d) 无效日期

invalidityDate 是非关键性 CRL Entry 扩展，指出知道密钥泄漏或怀疑泄漏的日期，即证书无效的日期。该日期必须比 CRL Entry 中的撤销日期要早。而 CRL Entry 中的撤销日期表明的是 CA 撤销证书的日期。无论是否可以获得此信息，都应该鼓励 CAs 将此信息与 CRL 用户共享。CAs 在 CRLs 中生成该扩展，取值的形式是 GeneralizedTime。

e) 证书发放者

certificateIssuer 和间接 CRL 一起使用，(间接 CRL，是 issuingDistributionPoint 扩展中有 indirectCRL 标识符的 CRL)。如果该扩展没有在间接 CRL 的第一个 Entry 中标出，那么该证书的发放者默认为 CRL 发放者。在间接 CRL 随后的 Entry 里，如果没有给出 certificateIssuer，则认为证书发放者与前一个 CRL Entry 的发放者相同。

f) CRL Entry 扩展使用总结

表 3 总结了 CRL Entry 扩展，表 4 总结了本标准中 CRL Entry 扩展的使用。

表 3 CRL Entry 扩展

扩展	使用	关键性与否
ReasonCode	说明证书撤销的原因	否
InstructionCode	和 certificateHold reasonCode 一起使用，表明遇到一个冻结证书时要采取的行为	否
InvalidityDate	证书无效的日期	否
CertficateIssuer	间接 CRL 中的撤销证书的颁发者	是

表 4 本标准中 CRL Entry 扩展的使用

扩展	CRL	客户
ReasonCode	支持；所有的 Entry 中均包括	可选—可以用来提供验证失败的信息
InstructionCode	不使用	可选
InvalidityDate	支持	可选—可以用来提供验证失败的信息
CertficateIssuer	可选	可选—如要支持间接 CRL 就必须
注：对 CRL 来说，“支持”意味着 CA 能够颁发含有该 CRL Entry 扩展的 CRL。对客户来说，“支持”意味着客户能够处理含有该 Entry 扩展的 CRL。		

6.4 证书认证路径

RFC 2459 中的第 6 章有具体定义，认证路径功能应由客户来实现。

6.5 事务消息格式

6.5.1 事务消息格式概述

本条给出了一套最小的支持 PKI 事务的消息格式。系统如果想实现这些事务就必须支持这些消息格式，并能够正确的产生、识别它们。这些消息格式以 ASN.1 的形式定义；消息利用 DER 编码传输。

6.5.2 全体 PKI 消息组件

a) PKI 消息

每个消息有四个组件：

```
PKIMessage ::= SEQUENCE {
header          PKIHeader,
body            PKIBody,
protection        [0]  PKIProtection      OPTIONAL,
extraCerts          [1]  SEQUENCE OF Certificate      OPTIONAL }
```

extraCerts 字段在本标准中用来放签名的证书序列，后面消息的描述中对这个字段不再具体描述。

b) PKI 消息头

所有的 PKI 消息都需要头信息来识别地址和处理。一些头信息会在传输特定包里给出。然而，如果 PKI 消息被签名，这个头信息也是被保护的。

下面是消息头的数据结构：

```
PKIHeader ::= SEQUENCE {
pvno                INTEGER             { ietf-version2 (1) },
sender              GeneralName,        -- 标识发送者
recipient           GeneralName,        -- 标识接收者
messageTime     [0] GeneralizedTime         OPTIONAL,
-- 该消息的产生时间(发送者使用)
-- 该时间对接收者来说仍然有意义。
protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
-- 计算保护比特的算法
senderKID       [2] KeyIdentifier           OPTIONAL,
recipKID        [3] KeyIdentifier           OPTIONAL,
-- 确定用来保护的特定密钥
transactionID   [4] OCTET STRING            OPTIONAL,
-- 确定事务，也就是说，相应的请求、响应和确认消息的 transactionID 相同。
senderNonce     [5] OCTET STRING            OPTIONAL,
recipNonce      [6] OCTET STRING            OPTIONAL,
-- nonces 用来提供重发保护，消息的产生者插入 senderNonce；该消息的接收者在相
-- 应消息中插入 recipNonce。
freeText        [7] PKIFreeText             OPTIONAL,
-- 这可以用来进行一些上下文说明(本字段的设计便于人工阅读)
generalInfo     [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue OPTIONAL
-- 用来传递上下文相关信息
-- (本字段不是主要为人工考虑的)}
PKIFreeText ::= UTF8String
```

头消息中的 transactionID 字段允许响应消息的接收者将它与请求相关联。一个 RA 或许在同一个时刻有很多请求未被处理。为了做到有效，从发送者的角度看，本字段的取值应当是独一无二的。

MessageTime 字段表明消息产生的时间。它的值必须是格林威治时间，必须精确到秒（即年月日时分秒 YYYYMMDDHHMMSSZ），即使秒是零。MessageTime 字段不包括分秒。

sender 和 recipient 字段定义为 GeneralName。系统必须支持 X.500 标识名和 RFC 822（Internet e-mail）标识名。

freetext 字段定义为 PKIFree Text，PKIFree Text 是 ASN.1 中的 UTF8String 型。UTF8String 是统一字符集，它包括了 ASCII，本标准中，PKIFree Text 仅包括 ASCII 中的字符。

ProtectionAlg 所有的签名消息，以及用消息认证码保护的消息都要有该字段。当一个消息不受保护时（即，在含有 PKIHeader 的 PKIMessage 中不出现 protection 字段时），ProtectionAlg 必须被忽略。

senderNonce 和 recipNonce 在 client 和 RA 中是不需要的；CA 如果收到了带有 senderNonce 的 PKIMessage，就必须在随后的消息里返回 recipNonce。本标准中，senderKID，recipKID，generalInfo 是不需要的。

c) PKI 消息正文

```
PKIBody ::= CHOICE {
-- message-specific body elements
cr      [2]   CertReqMessages,          --证书请求
cp      [3]   CertRepMessage,           --证书响应
p10cr   [4]   PKCS10CertReqMessages,    --引自[PKCS10]
rr      [11]  RevReqContent,            --撤销请求
rp      [12]  RevRepContent,            --撤销响应
conf    [19]  PKIConfirmContent,        --确认}
```

其他的 message-specific 正文元素在 RFC2510、RFC2511 中定义。本标准并不需要这些元素，所以这里把它们省略了。附录 C 中描述了 message-specific 正文元素的完整列表。

其他部分所说的 CertReq、CertRep、RevReq、RevRep 在 PKIMessage 里就是指 ir、ip、cr、cp、rr、rp。一个 PKCS#10 请求是指带有 p10cr 正文元素的消息。一个确认消息带有正文消息 conf。

d) PKI 消息保护

使用以下结构，所有 PKI 消息的完整性将得到保护：

```
PKIProtection::=BIT STRING
```

用来计算 PKIProtection 的输入是如下数据结构的 DER 编码：

```
   ProtectedPart::=SEQUENCE{
Header          PKIHeader,
body              PKIBody}
```

在大多数情况下，PKIProtection 字段将包含一个数字签名，PKIHeader 里的 protectionAlg 字段将包含一个 AlgorithmIdentifier，用来说明为保护消息所使用的数字签名算法。

在有些情况下，例如密钥更新，附加多重签名是必要的。这时，签名消息是嵌套的结构—每个已签名的消息成为一个 PKIBody 的元素；下一个签名应用于这个消息。这个处理将会不断重复直到所有的签名都已应用完毕。

6.5.3 通用数据结构

下面的数据类型对几个消息格式都是通用的。

a) 证书模板

在各种各样的 PKI 管理消息中,始发者可能会提供一些值来识别一个已存在的证书,或申请一个证书时在请求中含有一些值。CertTemplate 结构允许实体标明这些值。CertTemplate 包含作为一个证书所需的所有信息。

```
CertTemplate ::= SEQUENCE {
version      [0] Version                OPTIONAL,
-- 用来请求特殊语法版本
serial       [1] INTEGER                OPTIONAL,
-- 用来请求一个特殊的序列号或声明请求代表一个以前的证书持有者
signingAlg   [2] AlgorithmIdentifier    OPTIONAL,
subject      [3] Name                   OPTIONAL,
validity     [4] OptionalValidity       OPTIONAL,
issuer       [5] Name                   OPTIONAL,
publicKey    [6] SubjectPublicKeyInfo   OPTIONAL,
issuerUID    [7] UniqueIdentifier       OPTIONAL, -- not supported
subjectUID   [8] UniqueIdentifier       OPTIONAL, -- not supported
extensions   [9] Extensions             OPTIONAL,
-- 包含请求者希望在证书里具有的扩展
}
OptionalValidity ::= SEQUENCE {
notBefore    [0] Time OPTIONAL,
notAfter     [1] Time OPTIONAL
}
CertTemplates ::= SEQUENCE OF CertTemplate
```

如果 CertTemplate 出现,validity 字段将包含请求证书的请求发放日期(notbefore 字段)和期满日期(notAfter)。对 CertTemplate 中的 validity 字段和证书中的 validity 字段的解释相同。(见 5.2.1)

b) 签名私钥的拥有证明

若由主体产生密钥对,CA 需要验证证书请求中主体拥有对应于相应公钥的私钥。这是通过 ProofOfPossession 字段来实现的。

当用户申请一个签名证书,则用 ProofOfPossession 中的 signature 字段实现的,该 signature 是一个 POPOSigningKey 结构。该结构包含输入数据、一个算法标识符和一个签名。

输入数据作为 DER 编码的 popInput,它是由证书请求中的数据生成的。通常,它包含 CertTemplate 里的主体名和公钥。当一个新的主体请求通过公钥 mac 验证后,或是 RA 修改了主体名后,popInput 必须由 POPOSigningKey 结构中的可选数据和 CertTemplate 里的公钥构成。这就允许 RA 把拥有证明传递给 CA 而不用改变 CertTemplate。

```
ProofOfPossession ::= CHOICE {
raVerified                [0]  NULL,
-- 当 RA 已经验证了请求者拥有相应私钥时使用
  signature                    [1] POPOSigningKey,
-- 用户产生证书密钥对,并申请签名证书时使用
```

```
    keyEncipherment             [2] POPOPrivKey,
  -- 本标准不使用该域
    keyAgreement          [3] POPOPrivKey
  -- 本标准不使用该域}
    POPOSigningKey ::= SEQUENCE {
    poposkInput                 [0] POPOSKInput OPTIONAL,
    algorithmIdentifier                 AlgorithmIdentifier,
    signature                   BIT STRING }
  -- 对 popInput 的 DER 编码值签名(用"algorithmIdentifier")
  -- 注意：如果在 pop 字段里出现 poposkInput，popInput 就由 otherinput 构成。
  -- 如果 poposkInput 没有出现，主体就是 CertTemplate 中的名字。
  -- 注意 PopInput 的编码值是有意含糊的。
    PoposkInput ::= CHOICE {
    Subject name,
    Sender [0] generalName,
    publicKeyMAC [1] PKMACValue}
  -- pop 的计算是基于 popInput 的结构的。PopInput 的结构定义如下：
    PopInput ::= SEQUENCE {
    CHOICE {
    otherinput      popskInput,
    subject      name },
    publicKey      subjectpublicKey}
  -- 如果 poposkInput 在 pop 字段出现的话，popInput 就用 otherinpu 构成。如果 poposkInput
    不出现，主体就是 CertTemplate 中的名字。
  -- 注意 PopInput 的编码值是有意含糊的。
```

c) 证书请求消息

CertReqMsg 是证书请求的基本结构。CertReqMsg 由三个字段构成的 SEQUENCE，certReq；pop；regInfo(可选)。certReq 是类型 CertRequest，pop 包括了证明拥有私钥的信息，regInfo 包含了特定的注册过程信息。regInfo 字段是可选的。CertReqMsg 定义见附录 D。

```
CertReqMsg ::= SEQUENCE {
certReq         CertRequest,
pop             ProofOfPossession OPTIONAL,
-- 本标准规定加密密钥对必须集中产生，此时该字段不出现
-- 若集中产生签名密钥对，则该字段不出现
-- 若是由终端实体产生签名密钥对，则该字段必须出现
regInfo SEQUENCE SIZE(1..MAX) of AttributeTypeAndValue OPTIONAL }
```

CertRequest

CertRequest 的组成包括：请求标识符，证书模板，可选的控制信息。

```
CertRequest ::= SEQUENCE {
certReqId       INTEGER,            -- 用来匹配请求和相应的 ID
certTemplate    CertTemplate,       -- 选定的证书中的某些字段
controls        Controls OPTIONAL } -- 控制颁发的一些属性
```

certReqId 是一个 INTEGER，certTemplate 是 CertTemplate，controls 的定义如下。

CertReqMessages

CertReqMessages 是一个或多个 CertReqMessage 序列的结构。

CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMessage

本标准中,CertReqMessages 可以假设为仅由一个 CertReqMessage 构成的 SEQUENCE。即 MAX 可以定义为 1。

Controls

CertRequest 的产生者可以包括一个或多个适合请求处理的 control 值。

Controls ::= SEQUENCE SIZE (1..MAX) OF AttributeTypeAndValue

下列 control 在证书管理协议(CMP)里有定义:regToken;authenticator;pkiPublicationInfo;pkiArchiveOptions;oldCertId;protocolEncrKey。在具体实现规范中的事务时,这些 control 是可选的。这些 control 的 OIDs 如下。本标准不需要的 control (oldCertId, regToken, authenticator, pkiPublicationInfo, and pkiArchiveOptions)在这里没有描述。

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
dod(6) internet(1) security(5) mechanisms(5) pkix(7) }
-- arc for Internet X.509 PKI protocols and their components
id-pkip OBJECT IDENTIFIER ::{ id-pkix pkip(5) }
-- Registration Controls in CRMF
id-regCtrl OBJECT IDENTIFIER ::= { id-pkip regCtrl(1) }
id-regCtrl-oldCertId                OBJECT IDENTIFIER ::= { id-regCtrl 5 }
id-regCtrl-protocolEncrKey          OBJECT IDENTIFIER ::= { id-regCtrl 6 }
```

d) 协议加密密钥控制

如果存在的话,协议加密密钥控制明确了 CA 用来对 CertReqMessages 的响应进行加密的密钥。

当 CA 发送给申请者的信息需要加密的时候就可以使用这一控制。这些信息包括 CA 把产生的私钥发给申请者供其使用。

protocolEncrKey 控制由属性类型里的 id-reg-protocolEncrKey 的 OID 来确定。它的值的语法是 SubjectPublicKeyInfo.

e) PKI 消息状态码

所有的响应消息都包含一些状态信息。定义如下的取值:

```
PKIStatus ::= INTEGER {
granted                 (0),
--请求得到许可,没有变更
grantedWithMods         (1),
--请求得到许可,有变更;请求者负责确定变更。
rejection               (2),
--请求被拒绝
waiting                 (3),
--已收到请求但还未处理,处理后将会附加一个响应
revocationWarning       (4),
--给予警告:撤销已被请求,正在处理。
revocationNotification  (5),
--通知:已经撤销
keyUpdateWarning        (6) }
```

本标准并未用到 KeyUpdateWarning 状态码。

f) 失败信息

响应者使用以下句法提供更多失败信息。

```
PKIFailureInfo ::= BIT STRING { -- 因为会以多种方式失败!
badAlg                (0),      -- 不被识别或不被支持算法名
badMessageCheck       (1),      -- 完整性检查失败(例如,签名核对错误)
badRequest            (2),      -- 事务不被允许或支持
badTime               (3),      -- messageTime 与系统时间相差太多
badCertId             (4),      -- 证书标识名错
badDataFormat         (5),      -- 数据格式不对
wrongAuthority        (6),      -- 请求中指出的 authority 和响应者不同
incorrectData         (7),      -- 请求者的数据不对(在公证服务里用)
missingTimeStamp      (8),      -- 没有时戳,但是策略需要
badPoP                (9)       -- 拥有证明字段核实失败,
-- 需要更多的失败信息}
PKIStatusInfo ::= SEQUENCE {
status          PKIStatus,
statusString    PKIFreeText       OPTIONAL,
failInfo        PKIFailureInfo    OPTIONAL}
```

g) 确认协议

确认消息会在 PKIHeader 中带着所需的所有信息。因此,该数据结构内容为空。

```
PKIConfirmCotent::=NULL
```

h) 证书识别

为了识别特定的证书,采用了 CerId 结构

```
CerId::=SEQUENCE{
issuer      GeneralName,
serialNumber      INTEGER}
```

i) Centrally Generated Keys

本标准规定由第三方集中产生加密密钥对并通过带外方式提供给 CA,此时 CA 用 CertifiedKeyPair 结构发还证书。

```
CertifiedKeyPair ::= SEQUENCE {
certificate             [0] Certificate             OPTIONAL,
encryptedCert           [1] EncryptedValue          OPTIONAL,
privateKey              [2] EncryptedValue          OPTIONAL,
publicationInfo         [3] PKIPublicationInfo      OPTIONAL }
EncryptedValue ::= SEQUENCE {
intendedAlg             [0] AlgorithmIdentifier     OPTIONAL,
-- the intended algorithm for which the value will be used
symmAlg                 [1] AlgorithmIdentifier     OPTIONAL,
-- 加密这一 value 的对称算法
encSymmKey              [2] BIT STRING              OPTIONAL,
-- 用来加密这一 value 的对称密钥(已加密)
keyAlg                  [3] AlgorithmIdentifier     OPTIONAL,
```

-- 用来加密对称密钥的算法
valueHint [4] OCTET STRING OPTIONAL,
-- encValue 内容的简要描述或标识符
--(可能只对发送实体有意义,仅当将来发送实体对 EncryptedValue 重新检查时使用)
encValue BIT STRING
-- 加密的 value}

j) 带外信息

为了在带外传输一个 CA 的公钥,采用了 OOBCert 结构。OOBCert 只是一个 CA 的证书。

OOBCert::=Certificate

6.5.4 特殊操作的数据结构

a) 注册/证书请求

注册和证书请求消息(cr)包含一个 CertReqMessages 的数据结构,该结构规范了请求证书的特定取值。

CertReqMessages 是 CertReqMsg 结构的一个或多个序列。

CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

例外的,本标准里定义的事务假设 CertReqMessages 仅是一个 CertReqMsg 的 SEQUENCE。同时请签名证书和加密证书的可选事务请求要求 CertReqMessages 是大小为 2 的 SEQUENCE。本标准不鼓励同时申请两个证书,但不禁止。

b) 注册/证书响应

注册响应消息(cp)包含一个 CerRepMessage 结构(一个可选的公钥和一个 response)。特别的,本标准里定义的事务都假设响应仅是一个 CertResponse 的序列。一个例外就是含有签名证书和加密证书请求的事务;该事务的响应可以是响应消息的结合,即含有两个 CertResponse 的字段的序列。

CertResponse 包含一个请求 id、状态信息和一个 CertifiedKeyPair(可选)。CertifiedKeyPair 是一个包含四个可选字段的序列,四个可选字段分别是:一个证书,一个已加密证书、一个已加密私钥,以及公开信息。本标准中,证书字段总出现在 CertifiedKeyPair 里。仅当集中产生密钥管理密钥时才用到 PrivateKey,其余字段均不出现。

CertRepMessage ::= SEQUENCE {
caPub [1] Certificate OPTIONAL,
response SEQUENCE OF CertResponse}
CertResponse ::= SEQUENCE {
certReqId INTEGER, --与相应的请求相匹配
certRepStatus PKIStatusInfo,
certifiedKeyPair CertifiedKeyPair OPTIONAL;
—只有当状态为 granted 或 grantedWithMods 才存在
rspInfo OCTET STRING OPTIONAL
-- analogous to the id-regInfo-asciiPairs OCTET STRING defined
-- for regInfo in CertReqMsg [CRMF]}

如果 certRepStatus 包含 failInfo 字段,CertResponse 将不包含 certifiedKeyPair 并且 certRepStatus 字段的 status 的值将是 rejection. status 的值是 waiting 时,可选字段就都不出现。status 的值 revocationWarning 和 revocatonNotification 不会在消息中出现。

caPub 和 rspInfo 字段是不需要的,如果它们出现的话可以被忽略。本标准并不使用 CertifiedKeyPair 中的 encryptedCert 和 publicationInfo 字段。

c) 撤销请求的内容

当请求撤销一个证书时采用以下数据结构。请求者的名字在 PKIHeader 中显示。

```
RevReqContent ::= SEQUENCE OF RevDetails
RevDetails ::= SEQUENCE OF {
certDetails          CertTemplate,
--允许请求者尽可能多的指定所请求撤销的证书的参数(例如,不知道序列号时)
revocationReason     ReasonFlags,
-- 来自 DAM, 这样 CA 就知道该用哪个 Dist. point
badSinceDate         GeneralizedTime      OPTIONAL,
-- indicates best knowledge of sender
crlEntryDetails      Extensions
-- 要求的 crlEntryExtensions}
```

ReasonFlags 定义见附录 B。为清楚起见这里再说明一遍。

```
ReasonFlags ::= BIT STRING {
unused                (0),
keyCompromise         (1),
caCompromise          (2),
affiliationChanged    (3),
superseded            (4),
cessationOfOperation  (5),
certificateHold       (6),
removeFromCRL         (8) }
```

badSinceDate 和 revocationReason 表示的信息可以在 crlEntryDetails 中也同样表示出来,使用标准的 X.509 CRL 扩展 invalidityDate 和 reasonCode。推荐的位置是 crlEntryDetails,不过本标准建议 CA 处理其他位置的信息。当有冲突存在的时候,使用较早的日期。当撤销原因不匹配时,CA 应包括所有的原因。

d) 撤销响应内容

当对撤销请求做响应时,使用以下数据结构。把它发送给请求者。

```
RevRepContent ::= SEQUENCE {
status          PKIStatusInfo,
revCerts        [0] SEQUENCE OF CertId OPTIONAL,
-- 确定要请求撤销的证书
crls            [1] SEQUENCE OF CertificateList OPTIONAL,
-- 作为结果的相应的 RL}
```

在本标准中,revCerts 是一个或多个 CertID 字段的 SEQUENCE,crls 字段并不出现。

e) PKCS#10 证书请求

这个证书请求句法在 RFC 2314 中有定义,为清楚起见在此重复一遍。

```
PKCS10CertReqContent ::= SEQUENCE {
certificationRequestInfo      CertificationRequestInfo
signatureAlgorithm      SignatureAlgorithmIdentifier,
signature               Signature}
SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
Signature ::= BIT STRING
```

```
CertificationRequestInfo::= SEQUENCE {
version             Version,
subject             Name,
subjectPublicKeyInfo        SubjectPublicKeyInfo,
attributes          [0] IMPLICIT Attributes}
Version ::= INTEGER
Attributes ::= SET OF Attribute
```

Attributes 在 RFC 2985 中有定义。为了互操作而对 Attributes 的支持是可选的。即使出现了,也可以被忽略。

6.6 PKI 事务

6.6.1 PKI 事务概述

本条详细介绍了 PKI 的具体功能:注册请求、更新和撤销证书。也简要描述了访问目录服务的事务。

CA、RA 和证书持有者应能实现本条描述的所有事务。

对于 CA 和 RA 物理上在一起、不支持远端 RA 的 PKI 产品,CA 和 RA 之间的消息就可以忽略掉。

6.6.2 RA 发起的注册请求

RA 可以请求 CA 为一个终端实体颁发签名证书。这个事务分五步执行。第一步,终端实体通过带外事务(例如通过实际呈交一个磁盘)在一个签名消息中向 RA 提供一个公钥。第二步,RA 通过一个签名消息向 CA 请求证书。第三步,CA 通过一个含有证书或错误代码的签名消息回应 RA。第四步 RA 通过带外方式向终端实体提供 CA 的公钥和所颁发的证书(若是集中产生密钥对则还包括相应私钥),终端实体也可以直接从 CA 获得。第五步 RA 向 CA 发出确认消息。在这个过程中包括三条消息。

a) 从 RA 到 CA 的证书请求

RA 建立一个 PKIBody 为 cr 的 PKIMessage。其 PKIHeader 包括下列信息:

- pvno 是 1;
- transactionID 是该 RA 赋予这个事务的唯一整数;
- messageTime 是当前精确到秒的时间;
- sender 是 RA 的可辨别名;
- recipient 是 CA 的可辨别名;
- protectionAlg 是用来保护该消息的签名算法的算法标识符。

这个消息的消息体是 CertReqMessages,它是一个或多个 CertReqMessage 字段的序列。对于这个事务,CertReqMessages 是一个 CertReqMessage 的序列。这个 CertReqMessage 将包括如下信息:

- certReq 含有请求者希望包含在证书中的信息;
- pop 证明了对新证书私钥的拥有。

本标准中只支持终端实体产生签名密钥对,不支持终端实体产生加密密钥对。在进行签名私钥的拥有性证明时,如果由 RA 来实现,那么 RA 修改了主体名,popoSKInput 域出现,并且包含了原来的主体名。否则,RA 不修改主体名,那么 pop 域与请求者提交的主体名一致。详细原理见 6.5.3。

certReq 是 CertRequest,它是一个 certReqID、一个 CertTemplate 和 controls 的序列。对于这个事务,certReqID 可以是任何整数。

CertTemplate 将含有如下信息:

- version 是 v3(2);

- subject 是用户的名字；
- publicKey 提供了新证书的公钥，若用集中方式产生密钥对则不包括该字段；
- extensions 至少指定了与证书有关的证书策略的 OID。

如下的信息可以被包含在 CertTemplate 中：

- signingAlg 指定了首选签名算法；
- subject 指定了潜在证书持有者的可辨别名。

请求不应该包括如下信息：

- issuerUID；
- subjectUID。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 RA 的签名。

b) 从 CA 到 RA 的证书回应

CA 将返回以元素 cp 为 PKIBody 的 PKIMessage 给 RA。

PKIHeader 包括如下信息：

- pvno 是 1；
- transactionID 与 cr 消息中的 transactionID 相同；
- messageTime 是当前精确到秒的时间；
- sender 是 CA 的可辨别名；
- recipient 是 RA 的可辨别名；
- protectionAlg 是用来保护该消息的签名算法的算法标识符。

如果 senderNonce 在证书请求中被提供了，那么这个回应的消息头应该将其作为 recipNonce。PKIBody 元素 cp 是 CertRepMessage 类型。对于这个事务，CertRepMessage 将含有唯一的 response 字段。该 response 字段是一个 certReqId、status 和 certifiedKeyPair 的序列。如果 CA 颁发了一个证书，消息体将含有如下消息：

- certReqId 将与请求中的 certReqId 匹配；
- status 将是 granted 或者是 grantedWithMods；
- certifiedKeyPair 序列将至少含有一个字段——certificate，其将含有 GB/T 16264.8—2005 的证书。

证书必须满足如下性质：

- version 号应该是 v3(2)；
- publicKey 字段应该与证书请求中相同或者是由 CA 所产生的公钥；
- 主体可辨别名应该与证书请求中相同；
- 颁发者名字应该是 CA 的可辨别名；
- 如果 notBefore 出现在证书请求中，证书应该从颁发日和 notBefore 所指之日的较晚者之后生效；
- 如果 notAfter 出现在证书请求中，证书应该在该日或之前期满。

证书应该包括如下扩展(extensions)：

- subjectKeyIdentifier 域；
- 在 certificatePolicies 字段中至少包括一个证书策略的 OID；
- 包括 KeyIdentifier 字段的权威密钥标识符。

如果在证书请求消息中指定一个特定的密钥标识符，那么证书的 subjectKeyIdentifier 字段应该就是该密钥标识符。如果没有具体指明密钥标识符，CA 将用主体公钥的低 96 位 SHA-1 散列函数作为 subjectKeyIdentifier 字段中的 KeyIdentifier。散列函数是通过对证书中主体公钥字段的值(不包括标签和长度)进行计算得到的。

如果颁发者的证书或是 CRLs 不提供 X.500 目录服务，证书就应含有 issuerAltName 扩展中的 URLs 以及 CRLDistributionPoints 扩展中的 distributionPoint 字段。

如果 status 是 granted 和 grantedWithMods，那么 failInfo 字段可以不存在。

若 CA 拒绝了请求，则正文中包含以下信息：

- status 将是 rejected；
- failInfo 字段包含相应的错误码：
  - ◆ badAlg 说明 CA 不能验证签名，因为算法标识符无法识别或者不支持，
  - ◆ badMessageCheck 说明 PKIProtection 字段中的 mac 被检验，但是不匹配，
  - ◆ badPoP 说明 popoSigningKey 字段中的签名已经被检验，但是不匹配，
  - ◆ badRequest 说明响应者不允许或不支持该事务请求，
  - ◆ badTime 说明消息头中的 messageTime 字段中的时间与响应者的系统时间差别太大；

如果 status 是 rejected，那么 certifiedKeyPair 字段可以不出现。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 CA 的签名。

c) 确认消息

一旦接收到 cp，RA 应该产生 PKIConfirm 消息。确认消息的 PKIHeader 中的数据除了 messageTime 之外，应该与证书请求消息的 PKIHeader 中的数据相同。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 RA 的签名。

6.6.3 新实体的自我注册请求

一个新的实体如果从来没有从一个特定 CA 获得证书，他可以直接向那个 CA 申请一个新的证书。请求实体生成一个 PKI ir 消息来请求新证书，并在消息中包含和证书请求中公钥相对应的私钥的拥有证明。实体利用 RA 提供的一个秘密密钥和 SHA-1 HMAC 算法保护 PKI 消息。

如果 CA 接受自我注册请求，它将返回一个 ip 消息给证书持有者。该消息包含证书或者事务出错的原因代码。

a) RA 与实体的带外事务

自我注册申请证书的过程以 RA 和实体之间交换一个共享的秘密密钥开始。通过从共享的秘密中产生一个消息认证码，CA 能够对实体进行认证。

这里不指定这个带外事务的明确的内容和格式。但是秘密的密钥和可信 CA 的公钥信息应该以一种可信的方式传递给实体。

b) 从证书持有者到 CA 的自我注册请求

请求者建立一个 PKIBody 为 ir 的 PKIMessage。PKIHeader 包括如下信息：

- pvno 是 1；
- messageTime 是当前精确到秒的时间；
- sender 是请求者的可辨别名或一个电子邮件地址；
- recipient 是 CA 的可辨别名；
- protectionAlg 是用来保护消息的签名算法的算法标识符。

这个消息的消息体是 CertReqMessages，它是一个或多个 CertReqMessage 字段的序列。对于这个事务，CertReqMessages 是一个 CertReqMessage 的序列。这个 CertReqMessage 将包括如下信息：

- certReq 含有请求者希望包含在证书中的信息；
- popoSKInput 包含公钥的 mac 值；
- pop 证明了对证书私钥的拥有。

其中 pop 域通过与 CertTemplate 中的公钥相对应的私钥来产生，产生 pop 的输入数据包括

popoSKInput 中的公钥 mac 值和 CertTemplate 中的公钥。

certReq 是一个 CertRequest，它是一个 certReqID，一个 CertTemplate 和 controls 的序列。对于这个事务：

- certReqID 是任何整数；
- certTemplate 是一个 CertTemplate，其至少包括为新证书提供公钥的 publicKey 字段；
- controls 被省略。

CertTemplate 将含有如下信息：

- version 是 v3(2)；
- publicKey 提供了新证书的公钥。

如下的信息可以被包含在 CertTemplate 中：

- signingAlg 指定了首选签名算法；
- subject 指定了潜在证书持有者的可辨别名；
- extensions 至少指定了与证书有关的证书策略的 OID。

请求不应该包括如下信息：

- issuerUID；
- subjectUID。

PKIProtection 域包含一个请求者利用从 RA 获得的秘密产生的值。实体利用 RA 提供的秘密密钥产生一个 96bit 的 SHA1-HMAC。protectionAlg 域应该设成 SHA1-HMAC，PKIProtection 的值应该是 96 位的消息认证码。计算 PKIProtection 的输入是如下数据结构的 DER 编码：

ProtectedPart::=SEQUENCE {

PKIHeader,

PKIBody}

c) 从 CA 到证书请求者的自我注册请求的回应

CA 将返回给证书持有者一个 PKIBody 为 ip 的 PKIMessage 消息。PKIHeader 包含如下信息：

- pvno 是 1；
- messageTime 是当前精确到秒的时间；
- sender 是 CA 的可辨别名；
- recipient 是证书请求消息头中 sender 域的值；
- protectionAlg 是用来保护消息的签名算法的算法标识符。

如果 ir 消息中提供了 transactionID，这个回应的消息头中将包括同样的 transactionID。如果 ir 消息中提供了 senderNonce，这个回应的消息头中将包含 recipNonce。

PKIBody 元素 ip 是 CertRepMessage 类型。如果 CA 颁发了一个证书，消息体将含有如下信息：

- status 将是 granted 或者是 grantedWithMods；
- certificate 包含新的 GB/T 16264.8—2005 的证书。

如果 status 是 granted 或者是 grantedWithMods，failInfo 域可以不存在。

如果 CA 拒绝了请求，消息体将含有如下信息：

- status 将是 rejected；
- failInfo 包含适当的错误代码：
  - ◆ badAlg 说明 CA 不能验证签名，因为算法标识符无法识别或者不支持，
  - ◆ badMessageCheck 说明 PKIProtection 字段中的 mac 被检验，但是不匹配，

◆ badPoP 说明 popoSigningKey 字段中的签名已经被检验，但是不匹配，

◆ badRequest 说明响应者不允许或不支持该事务请求，

◆ badTime 说明消息头中的 messageTime 字段中的时间与响应者的系统时间差别太大，

◆ badCertID 表明 CertDetails 中的信息无法确定一个未过期、未撤销的证书；

如果 status 是 rejected，certificate 域可能不存在，如果存在证书应该遵从于 6.2.1 中定义的证书轮廓。

证书应该包括如下扩展(extensions)：

- subjectKeyIdentifier 域；
- 在 certificatePolicies 字段中至少包括一个证书策略的 OID；
- 包括 KeyIdentifier 字段的权威密钥标识符。

如果在 ir 消息中指定一个特定的密钥标识符，那么证书的 subjectKeyIdentifier 字段应该就是该密钥标识符。如果没有具体指明密钥标识符，CA 将用主体公钥的低 96 位 SHA-1 散列函数作为 subjectKeyIdentifier 字段中的 KeyIdentifier。散列函数是对证书中主体公钥字段的值(不包括标签和长度)计算得到的。

如果 ir 消息中包含了除 subjectKeyIdentifier 之外的其他扩展，CA 将修改或者忽略该扩展。

如果颁发者的证书或是 CRLs 不提供 X.500 目录服务，证书就应含有 issuerAltName 扩展中的 URLs 以及 CRLDistributionPoints 扩展中的 distributionPoint 字段。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 CA 的签名。

d) 确认消息

一旦接收到 ip，证书持有者应该产生一个 PKIConfirm 消息。确认消息的 PKIHeader 中的数据除了 messageTime 之外，应该与证书请求消息的 PKIHeader 中的数据相同。

如果请求被接受，PKIProtection 域包含证书持有者的数字签名，利用与新的签名证书对应的私钥对消息头和消息体的 DER 编码序列计算签名。如果请求被拒绝，PKIProtection 域包含 SHA-1 HMAC，利用共享秘密对消息头和消息体的 DER 编码计算。

一旦接收到 PKIConfirm，CA 应该产生一个 PKIConfirm 消息。确认消息的 PKIHeader 中的数据除了 messageTime 之外，应该与证书请求消息的 PKIHeader 中的数据相同。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 CA 的签名。

6.6.4 已知实体的自我注册请求

一个实体不是当前的证书持有者，但是以前曾经从一个特定 CA 获得过证书，他可以直接向那个 CA 申请一个新的证书。请求实体生成一个 PKI cr 消息来请求新证书，并在消息中包含与证书请求中公钥所对应的私钥的拥有证明。实体利用 RA 提供的一个秘密密钥和 SHA-1 HMAC 算法保护 PKI 消息。

如果 CA 接受自我注册请求，它将返回一个 cp 消息给证书持有者。该消息包含证书或者事务出错的原因代码。

a) RA 与实体的带外事务

自我注册申请证书的过程以 RA 传递一个共享的秘密密钥给实体开始。通过从共享的秘密中产生的一个消息认证码，CA 能够对实体进行认证。

这里不指定这个带外事务的明确的内容和格式。但是秘密的密钥和可信 CA 的公钥信息应该以一种可信的方式传递给实体。

b) 从证书持有者到 CA 的自我注册请求

请求者建立一个 PKIBody 为 cr 的 PKIMessage。PKIHeader 包括如下信息：

- pvno 是 1；

- messageTime 是当前精确到秒的时间；
- sender 是请求者的可辨别名或一个电子邮件地址；
- recipient 是 CA 的可辨别名；
- protectionAlg 是用来保护消息的签名算法的算法标识符。

这个消息的消息体是 CertReqMessages，它是一个或多个 CertReqMessage 字段的序列。对于这个事务，CertReqMessages 是一个 CertReqMessage 的序列。这个 CertReqMessage 将包括如下信息：

- certReq 含有请求者希望包含在证书中的信息；
- popoSKInput 包含公钥的 mac 值；
- pop 证明了对证书私钥的拥有。

其中 pop 域通过与 CertTemplate 中的公钥相对应的私钥来产生，产生 pop 的输入数据包括 popoSKInput 中的公钥 mac 值和 CertTemplate 中的公钥。

certReq 是一个 CertRequest，它是一个 certReqID，一个 CertTemplate 和 controls 的序列。对于这个事务：

- certReqID 是任何整数；
- certTemplate 是一个 CertTemplate，其至少包括为新证书提供公钥的 publicKey 字段。

CertTemplate 将含有如下信息：

- version 是 v3(2)；
- publicKey 提供了新证书的公钥。

如下的信息可以被包含在 CertTemplate 中：

- signingAlg 指定了首选签名算法；
- subject 指定了潜在证书持有者的可辨别名。
- extensions 至少指定了与证书有关的证书策略的 OID。

请求不应该包括如下信息：

- issuerUID；
- subjectUID。

PKIProtection 域包含一个请求者利用从 RA 获得的秘密产生的值。实体利用 RA 提供的秘密密钥产生一个 96bit 的 SHA1-HMAC。protectionAlg 域应该设成 SHA1-HMAC，PKIProtection 的值应该是 96 位的消息认证码。计算 PKIProtection 的输入是如下数据结构的 DER 编码：

ProtectedPart::=SEQUENCE {
PKIHeader,
PKIBody}

c) 从 CA 到证书请求者的自我注册请求的回应

CA 将返回给证书持有者一个 PKIBody 为 cp 的 PKIMessage 消息。PKIHeader 包含如下信息：

- pvno 是 1；
- messageTime 是当前精确到秒的时间；
- sender 是 CA 的可辨别名；
- recipient 是证书请求消息头中 sender 域的值；
- protectionAlg 是用来保护消息的签名算法的算法标识符。

如果 cr 消息中提供了 transactionID，这个回应的消息头中将包括同样的 transactionID。如果 cr 消息中提供了 senderNonce，这个回应的消息头中将包含 recipNonce。

PKIBody 元素 cp 是 CertRepMessage 类型。如果 CA 颁发了一个证书，消息体将含有如下信息：

- status 将是 granted 或者是 grantedWithMods；
- certificate 包含新的 GB/T 16264.8—2005 的证书。

如果 status 是 granted 或者是 grantedWithMods，failInfo 域可以不存在。

如果 CA 拒绝了请求，消息体将含有如下信息：

- status 将是 rejected；
- failInfo 包含适当的错误代码：
  - ◆ badAlg 说明 CA 不能验证签名，因为算法标识符无法识别或者不支持，
  - ◆ badMessageCheck 说明 PKIProtection 字段中的 mac 被检验，但是不匹配，
  - ◆ badPoP 说明 popoSigningKey 字段中的签名已经被检验，但是不匹配，
  - ◆ badRequest 说明响应者不允许或不支持该事务请求，
  - ◆ badTime 说明消息头中的 messageTime 字段中的时间与响应者的系统时间差别太大，
  - ◆ badCertID 表明 CertDetails 中的信息无法确定一个未过期、未撤销的证书；

如果 status 是 rejected，certificate 域可能不存在，如果存在证书应该遵从于 6.2.1 中定义的证书轮廓。

证书应该包括如下扩展(extensions)：

- subjectKeyIdentifier 域；
- 在 certificatePolicies 字段中至少包括一个证书策略的 OID；
- 包括 KeyIdentifier 字段的权威密钥标识符。

如果在 cr 消息中指定一个特定的密钥标识符，那么证书的 subjectKeyIdentifier 字段应该就是该密钥标识符。如果没有具体指明密钥标识符，CA 将用主体公钥的低 96 位 SHA-1 散列函数作为 subjectKeyIdentifier 字段中的 KeyIdentifier。散列函数是对证书中主体公钥字段的值(不包括标签和长度)计算得到的。

如果 cr 消息中包含了除 subjectKeyIdentifier 之外的其他扩展，CA 将修改或者忽略该扩展。

如果颁发者的证书或是 CRLs 不提供 X.500 目录服务，证书就应含有 issuerAltName 扩展中的 URLs 以及 CRLDistributionPoints 扩展中的 distributionPoint 字段。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 CA 的签名。

d) 确认消息

一旦接收到 cp，证书持有者应该产生一个 PKIConfirm 消息。确认消息的 PKIHeader 中的数据除了 messageTime 之外，应该与证书请求消息的 PKIHeader 中的数据相同。

如果请求被接受，PKIProtection 域包含证书持有者的数字签名，利用与新的签名证书对应的私钥对消息头和消息体的 DER 编码序列计算签名。如果请求被拒绝，PKIProtection 域包含 SHA-1 HMAC，利用共享秘密对消息头和消息体的 DER 编码计算。

一旦接收到 PKIConfirm，CA 应该产生一个 PKIConfirm 消息。确认消息的 PKIHeader 中的数据除了 messageTime 之外，应该与证书请求消息的 PKIHeader 中的数据相同。

PKIProtection 字段含有根据消息头和消息体的 DER 编码序列计算的 CA 的签名。

6.6.5 证书更新

拥有当前有效(指在有效期内、未被撤销)证书的 PKI 实体可以直接向该证书的签发 CA 要求签发一份新的证书。PKI 实体生成 PKI kr(Key update Request 密钥更新申请)消息，信息中包括了证书申请和相应的 pop。然后 PKI 实体使用有效证书的对应私钥对 PKI kr 消息进行签名。

如果 CA 的 CPS 支持证书更新，则 CA 返回 kp(Key update response 密钥更新响应)消息。kp 消

息包含了新生成的证书或者是事务失败的代码。

如果新证书成功生成，则可能还有两个可选的消息。分别是：PKI 实体在收到新的证书后，给 CA 发出确认；然后 CA 也可回应一个确认消息。

a) 从证书持有者到 CA 的证书更新申请

证书持有者生成 PKIBody 是 kr 的 PKIMessage，PKIMessage 中的 PKIHeader 包括了如下信息：

- pvno 是 1；
- messageTime 是精度到秒的当前时间；
- sender 是证书持有者的可辨别名；
- recipient 是 CA 的可辨别名；
- protectionAlg 是用于保护消息签名算法 ID。

消息体是 CertReqMessages，是一个或者多个 CertReqMessage 组成的序列。对于证书更新事务，CertReqMessages 只能是包括了一个 CertReqMessage 的序列。在消息 CertReqMessage 中包括了如下信息：

- certReq 包含了申请者要求包括在证书中的各种信息；
- pop 是新证书公钥的对应的 pop 证明。

pop 必须是由 publicKey 域的公钥对应的私钥产生的。

certReq 是 CertRequest，CertRequest 是由一个 certReqID、一个 certTemplate 和 controls 所组成的序列。对于证书更新事务：

- version 必须是 v3(2)；
- publicKey 中是新证书的公钥。

CertTemplate 中可能包括了如下信息：

- signingAlg 指明了首选的签名算法；
- subject 则是预期的证书持有者的可辨别名。

如果消息中没有 signingAlg，那么 CA 必须使用终端实体的公钥对应的算法签名。

申请中不应该包括如下信息：

- issuerUID；
- subjectUID。

PKIProtection 域是使用当前有效证书的对应私钥对消息头和消息体的 DER 编码信息的签名结果。

b) 从 CA 到证书持有者的证书更新响应

CA 生成 PKIBody 是 kp 的 PKIMessage，PKIMessage 中的 PKIHeader 包括了如下信息：

- pvno 是 1；
- messageTime 是精度到秒的当前时间；
- sender 是 CA 的可辨别名；
- recipient 是证书持有者的可辨别名，也就是 kr 消息的 sender；
- protectionAlg 是用于保护消息签名算法 ID。

如果在 kr 消息中具有 transtionID，则 CA 回应消息的消息头中包括同样的 transtionID。如果在 senderNonce 消息中具有 senderNonce，则 CA 回应消息的消息头中应该包括同样的 recipNonce。

在 PKIBody 中，是 kp 元素。kp 是 CertRepMessage 格式。如果 CA 签发了新证书，在消息体中应包括如下信息：

- status 是 granted 或者 grantedWithMods；

- certificate 是新生成的 GB/T 16264.8—2005 证书。

新生成的证书应该包括如下扩展：

- subjectKeyIdentifier 域；
- 在 certificatePolicies 字段中至少包括一个证书策略 OID；
- 在 KeyIdentifier 域中有一个机构密钥标识符。

certificatePolicies 扩展应该和当前有效证书一致。如果在 kr 消息中包含了密钥标识符信息，则证书中应该将其包含在 subjectKeyIdentifier；如果没有提供密钥标识符信息，则 CA 应该对证书公钥信息计算低 96 bit 的 SHA-1 摘要值作为 subjectKeyIdentifier 的值。摘要值是对证书中的主体公钥进行计算的，不包括标签和长度。

如果 kr 信息中包括了 subjectKeyIdentifier 之后的扩展，则 CA 可以对其他的扩展信息进行修改或者忽略。

如果发布者的证书和 CRL 不能够从已知的 X.500 目录中得到，则新证书中应该在 issuerAltName 中包括 URLs 信息，在 CRLDistributionPoints 控制中包括 distributionPoint 域。

如果 status 是 granted 或者 grantedWithMods，则 failInfo 域不存在。

如果 CA 拒绝用户的申请，消息体包括如下信息：

- status 是 rejected；
- failInfo 包括了合适的代码，可以是：
  - ◆ badAlg 表示 CA 不支持或者不认识 kr 指定的签名算法，
  - ◆ badPop 表示 popoSigningKey 域中的签名有误，
  - ◆ badMessageCheck 表示 PKIProtection 域中的签名有误，
  - ◆ badRequest 表示回应者不支持或者不允许该事务，
  - ◆ badTime 表示消息头中的 messageTime 的时间和回应者的系统时间相差太大，
  - ◆ badCertId 表示证书序号不能是 serial 域中所填的非零值；

如果 status 是 rejected，certifcate 域不存在。

PKIProtection 域包含了 CA 对于消息头和消息体的 DER 编码的签名。

c) 确认消息

收到 kp 的时候，证书持有者应该产生 PKIConfirm 消息。PKIHeader 中的信息和利用 RA 向 CA 申请证书的情况下的 PKIHeader 一致，除了 messageTime 是当前时间。

如果证书更新申请被接受，则 PKIProtection 是证书持有者使用新证书的对应私钥，对于消息头和消息体的 DER 编码的签名。如果申请被拒绝，则 PKIProtection 是证书持有者使用当前有效证书的对应私钥，对于消息头和消息体的 DER 编码的签名。

收到 PKIConfirm 的时候，CA 应该产生 PKIConfirm 消息。PKIHeader 中的信息和 kp 一致，除了 messageTime 是当前时间。

PKIProtection 是 CA 对于消息头和消息体的 DER 编码的签名。

6.6.6 PKCS#10 自我注册请求

目前不是证书持有者的实体可以用 RFC 2314 定义的证书请求语句直接向 CA 申请证书。请求实体产生 PKCSReq 类型的 PKIMessage 消息来申请证书，在证书请求中的正文里包括相应私钥的拥有证明，并用 RA 以带外方式提供的密钥来加密该 PKIMessage.

CA 返回一个证书请求响应消息给证书申请者。该消息将包含证书或处理失败原因代码。

证书请求者和 RA 之间需用带外方式完成一个秘密消息的交换。

a) 从证书持有者到 CA 的自我注册请求

请求者产生一个包含 PKIBody 元素 p10cr 的 PKIMessage，PKIHeader 包含以下信息：

- pvno 是 1；

- messageTime 当前精确到秒时间；
- sender 请求者的可辨别名或电子邮件地址；
- recipient CA 的可辨别名；
- protectionAlg 用来保护该消息的签名算法的算法标识符。

PKIBody 是 PKCS10CertReqMessages 类型，此类型是一个 certificationRequestInfo、signatureAlgorithm 和 signature 的序列。certificationRequestinfo 包含以下信息：

- version 是 V3(2)；
- subject 当且仅当 serial 为零时才出现，指明证书持有者的可辨别名；
- subjectPublicKeyInfo 指明了新证书的公钥和相应的算法标识符。

signatureAlgorithm 字段包含用来生成 signature 字段的私钥的相应算法标识符；签名是对 DER 编码后的 certficationRequestInfo 进行的。

PKIProtection 字段包含一个请求者用从 RA 获得的秘密信息生成的一个值。实体用 RA 提供的密钥产生一个 MAC。protectionAlg 字段应为所用的消息认证算法的 OID，并且 PKIProtection 的值就是消息认证码。PKIProtection 的输入是如下数据结构的 DER 编码：

ProtectedPart：：=SEQUENCE{

header　　PKIHeader，

body　　PKIBody}

b) 从 CA 到证书申请者的 PKCS 证书请求响应

CA 会返回一个包含 PKIBody 元素 cp 的 PKIMessage 给证书持有者。PKIHeader 包含以下信息：

- pvno 是 1；
- messageTime 当前精确到秒的时间；
- sender CA 的可辨别名；
- recipient 证书请求头消息中的 sender 的值；
- protectionAlg 用来保护该消息的签名算法的算法标识符。

如果 PKCSReq 中有 transactionID，则响应的 PKIHeader 中应包含相同的 transactionID。

PKIBody 的元素是一个 CerRepMessage 类型的 cp。若 CA 发放了证书，PKIBody 将包含以下信息：

- status 为 granted 或 grantedWithmods，
- certificate 包含新的 GB/T 16264.8—2005 证书。

若 p10cr 消息中指定了特定的密钥标识符，证书将在 subjectKeyIdentifier 字段包含该密钥标识符。如果没有指明具体的密钥标识符，CA 将用主体公钥的低 96 位 SHA-1 散列函数作为 subjectKeyIdentifier 字段中的 KeyIdentifier。散列函数是对证书中主体公钥字段的值(不包括标签和长度)计算得到的。

如果 status 为 granted 或 grantedWithmods，failInfo 将不出现；若 CA 拒绝请求，正文将包含以下信息：

- status 为 rejected；
- failInfo 包含相应的错误代码：
  - ◆ badAlg 指明算法的标识符不被识别或不被支持，因此 CA 无法验证签名，
  - ◆ badpop 指明 pk10cr 的 signature 字段的签名被校验，但是不匹配，
  - ◆ badMessageCheck 指明 PKIMessage 的 PKIProtection 字段里的 MAC 被拒绝，
  - ◆ badrequest 指明响应者不允许或不支持请求，
  - ◆ badtime 指明消息头中的 messageTime 字段中的时间与响应者的系统时间差别

太大；

若 status 字段为 rejected，certifitate 字段将不出现。

证书还要包括以下扩展：

- 一个 subjectKeyIdentifier 字段；
- 在 certificatePolicies 字段中至少包括一个证书策略 OID；
- 一个 authority 密钥标识符，包括一个 KeyIdentifier 字段。

若 cr 消息中还包括除 subjectKeyIdentifier 以外的其他扩展，CA 就可以修改或忽略请求扩展。

如果颁发者的证书或是 CRLs 不提供 X.500 目录服务，证书就应含有 issuerAltName 扩展中的 URLs 以及 CRLDistributionPoints 扩展中的 distributionPoint 字段。

PKIProtection 字段包含 CA 的签名，在消息头和消息体的 DER 编码序列上签名。

6.6.7 撤销请求

证书持有者可以请求撤销自己的证书。证书持有者产生一个 RevReq 消息，对该消息进行签名并发送给相应 RA，并在 RA 审查通过用户的身份后向 CA 发出相应撤销信息。该签名必须用未过期、未被撤销的签名证书的相应私钥产生(可以是要撤销的证书)。RevReq 消息要标识出想撤销的证书以及要撤销的原因。CA 回应 RA 一个 RevRep 消息，RA 再回应证书持有者相应的 RevRep 消息。

如果消息 rr (RevReq)中包含 transactionID，则 CA 和 RA 所响应的 rp (RevRep) 消息中也应包含相同的 transactinID，注意其中从证书持有者所发出的 rr 和 RA 所发出的 rr 消息中的 transactinID 可以不同。rp 消息至少要包含 status 字段以反映请求的状态和 revDetails 字段以表示欲撤销的证书。

a) 从证书持有者到 RA 的撤销请求

证书持有者生成一个包含一个 PKIBody 元素 rr 的 PKIMessage。PKIHeader 包含以下信息：

- pvno 是 1；
- transactionID 与终端实体名一起唯一标识一个终端实体与 RA 之间事务的整数；
- messageTime 为当前精确到秒的时间；
- sender 为证书持有者的可辨别名；
- recipient 为 RA 的可辨别名；
- protectionAlg 为保护消息而使用的签名算法标识符。

消息的正文 PKIBody 为 RevReqContent，RevReqContent 是 RevDetails 的序列。RevDetails 是由 CertDetails 和三个可选字段组成的序列：原因标志；怀疑或丢失的日期和时间；以及 crlEntryDetails(一个 CRL Entry 扩展的序列)。CertDetails 被定义为一个 CertTemplate。在本标准中，RevReqContent 是一个 RevDetails 的序列。CertDetails，最少包括以下信息：

- serial 证书序列号；
- issuer 证书发放者的标识名。

或是

- subject 证书持有者的标识名；
- issuer 证书发放者的标识名。

CertDetails 还可在 extensions 字段中包含一个 subjectKeyIdentifier。(如果请求者希望撤销颁发给某个主体的所有证书，CertDetails 就应仅含有 subject 和 issuer。也就是说，仅希望撤销单个证书的请求就只含有相应的序列号或是 subjectKeyIdentifier)。

RevDetails 要包括带有 reasonCode 扩展的 crlEntryDetails，也可以包括 invalidityDate 扩展来说明何时该证书作废。原因代码也可以不是 removeFromCRL。

PKIProtection 字段含有请求者的签名，即对头和正文的 DER 编码进行签名。终端实体要用相应 CA 所颁发的当前有效签名证书的相应私钥进行签名。

b) 从 RA 到 CA 的撤销请求

RA 或证书持有者生成一个包含一个 PKIBody 元素 rr 的 PKIMessage。PKIHeader 包含以下信息：

- pvno 是 1；
- transactionID 标识 RA 名一起唯一标识一个 RA 与 CA 之间事务的整数；
- messageTime 为当前精确到秒的时间；
- sender 为 RA 的可辨别名；
- recipient 为 CA 的可辨别名；
- protectionAlg 为保护消息而使用的签名算法标识符。

消息体与从证书持有者到 RA 的撤销请求相同。

PKIProtection 字段含有 RA 的签名，即对头和正文的 DER 编码进行签名。RA 要用相应 CA 所颁发的当前有效签名证书的相应私钥进行签名。

c) 从 CA 到 RA 的撤销响应

CA 返回一个含有 PKIBody 元素 rp 的 PKIMessage 给请求者。PKIHeader 包含以下信息：

- pvno 是 1；
- transactionID 和从 RA 到 CA 的撤销请求 rr 消息中的 transactionID 字段一样；
- messageTime 为当前精确到秒的时间；
- sender 为 CA 的可辨别名；
- recipient 为 RA 的可辨别名；
- protectionAlg 为保护消息而使用的签名算法标识符。

如果相应从 RA 到 CA 的撤销请求中有 senderNonce ，则响应的 PKIHeader 中应把它作为 recipNonce 。

PKIBody 是 RpContent，如果 CA 撤销了证书，正文将包含以下信息：

- status 是 granted 或是 grantedWithMods；
- revDetails 将包含已撤销证书的 CertId ；
- 如果 status 是 granted 或 grantedWithMods，failInfo 字段也可以不出现。

如果 CA 拒绝了请求，正文就要包括如下信息：

- status 为 rejected；
- failInfo 包含相应的失败代码：
  - ◆ badAlg 指明 CA 不能验证签名，因为算法的标识符不被识别或不被支持，
  - ◆ badMessageCheck 指明 PKIProtection 字段里的签名得到验证，但是不匹配，
  - ◆ badRequest 字段指明响应者不允许或不支持请求，
  - ◆ badTime 字段指明消息头中的 messageTime 字段中的时间与响应者的系统时间差别太大，
  - ◆ badCertID 表明 CertDetails 中的信息无法确定一个未过期、未撤销的证书；

对于能够确定有问题的证书，revDetails 将包含这个被拒绝撤销证书的 CertId 。PKIProtection 字段包含 CA 的签名，即对头和正文的 DER 编码进行签名。若 CA 生成 CRLs，并且撤销请求被接受，CRL 将有以下值：

- userCertificate 字段中的被撤销证书的序列号；
- revocationDate 收到撤销请求的日期和时间；
- crlEntryExtensions 要出现并包括：
  - ◆ RevDetails 字段中的 reasonCode ，除非 CA 的策略有专门规定，
  - ◆ （可选的）RevDetails 字段中的 badSinceDate 扩展可以是 invalidaityDate。

d) 从 RA 到证书持有者的撤销响应

RA 在收到 CA 的回应消息后，返回一个含有 PKIBody 元素 rp 的 PKIMessage 给证书持有者。

PKIHeader 包含以下信息：

- pvno 是 1；
- transactionID 和从 RA 到 CA 的撤销请求 rr 消息中的 transactionID 字段一样；
- messageTime 为当前精确到秒的时间；
- sender 为 CA 的可辨别名；
- recipient 为 RA 的可辨别名；
- protectionAlg 为保护消息而使用的签名算法标识符。

如果相应的从证书持有者到 RA 的撤销请求消息中有 senderNonce，则响应的 PKIHeader 中应把它作为 recipNonce。

PKIBody 是 RpContent，内容与从 CA 到 RA 的撤销响应相同，PKIProtection 字段包含 RA 的签名，即对头和正文的 DER 编码进行签名。

6.6.8 集中产生密钥对和密钥管理证书申请

拥有当前有效证书的 PKI 实体可以向该证书的签发 CA 提出申请，申请产生加密密钥对并签发相应的证书。下面以 RSA 为例说明申请过程，在国内应用时，应使用国家密码管理主管部门审核批准的相关算法。发出申请的实体：

- 产生临时的密钥管理密钥；
- 生成 PKI cr(证书申请 certificate request)消息，申请 RSA 密钥管理证书，cr 消息中包括了上一步的临时密钥；
- 利用当前有效证书的对应私钥，对消息进行签名；
- 发送给 CA。

如果 CA 的 CPS 支持集中产生加密密钥对，则 CA 执行如下操作：

- CA 按申请消息的要求产生密钥对，签发密钥管理证书；
- 申请消息使用临时 RSA 密钥；
- CA 产生对称密钥；
- 利用对称密钥加密新产生的私钥；
- 使用临时 RSA 公钥加密对称密钥；
- 产生和返回 cp(certificate response 证书响应)消息给证书持有者。cp 消息中包括了新生成的证书和加密后的私钥，或者是事务失败的代码。

a) 集中产生密钥对申请

证书持有者产生证书申请消息：PKIMessage 消息中的 PKIBody 部分是 cr 元素。PKIHeader 包括了如下信息：

- pvno 是 1；
- messageTime 是当前时间，精确到秒；
- sender 是证书持有者的可辨别名；
- recipient 是 CA 的可辨别名；
- protectionAlg 是用于保护消息签名算法 ID。

消息体是 CertReqMessages，是一个或者多个 CertReqMessage 组成的序列。对于集中产生密钥对申请和密钥管理证书事务，CertReqMessages 只能是包括了一个 CertReqMessage 的序列。在消息 CertReqMessage 中包括了如下信息：

- certReq 包含了申请者要求包括在证书中的各种信息。

certReq 是 CertRequest，CertRequest 是由一个 certReqID、一个 certTemplate 和 controls 所组成的序列。对于集中产生密钥对申请和密钥管理证书事务：

- certReqID 是任意整数；
- certTemplate 是一个 CertTemplate；
- controls 包含了 protocolEncrKey 控制。protocolEncrKey registration control 的值是临时公钥的值和参数(如果需要)，以及算法 OID。

CertTemplate 包含了如下信息：

- version 必须是 v3(2)；
- publicKey 指明了算法和可选的参数，说明了所申请的密钥。

CertTemplate 中可能包括了如下信息：

- signingAlg 指明了首选的签名算法。

如果没有 signingAlg，则 CA 应该使用 protectionAlg 中说明的签名算法。

申请消息里面不应该包括如下信息：

- issuerUID；
- subjectUID。

PKIProtection 域是使用当前有效证书的对应私钥对消息头和消息体的 DER 编码信息的签名结果。

b) 集中产生密钥对回应

CA 返回密钥更新(PKIMessage，PKIBody 是 cp 元素)消息给证书持有者。

PKIHeader 包括了如下信息：

- pvno 是 1；
- messageTime 是当前时间，精度到秒；
- sender 是 CA 的可辨别名；
- recipient 是证书持有者的可辨别名，也就是 cr 消息的 sender；
- protectionAlg 是用于保护消息签名算法 ID。

如果在 cr 消息中具有 transtionID，则 CA 回应消息的消息头中包括同样的 transtionID。如果在 senderNonce 消息中具有 senderNonce，则 CA 回应消息的消息头中应该包括同样的 recipNonce。

PKIBody 是 cp 元素。也就是 CertRepMessage 格式。如果 CA 签发了新证书，在消息体中应包括如下信息：

- status 是 granted 或者是 grantedWithMods；
- certificate 包括了新签发的 GB/T 16264.8—2005 证书；
- certifiedKeyPair 必须存在。

certifiedKeyPair 将在 certOrEncCert 域中包括证书，在 privateKey 域中包括加密过的私钥。申请者使用临时 RSA 公钥，privateKey 域应该包括 symmAlg、encSymmKey 和 encValue。

- symmAlg 应该包括 tDEA-ecb 的 OID，keyingOption 应该是 option-2(表示 K1 和 K2 是独立产生的，K3 = K1)；
- encSymmKey 是对 CA 产生的对称密钥的加密计算结果，使用临时 RSA 公钥进行加密；
- 对称密钥 K1 和 K2，应串接为 TwoKeys，见(6.2.2.5)；
- TwoKeys 应该按照 RFC 2313 说明的 RSA 加密方法进行加密；
- 加密结果编码为 encSymmKey。加密结果编码为 BIT STRING，BIT STRING 的最高位就是加密结果的最高位。

新的用户证书应该包括如下扩展：

- subjectKeyIdentifier 域；
- 在 certificatePolicies 字段中至少包含一个证书策略 OID；
- 在 KeyIdentifier 域中有一个机构密钥标识符。

CA 应该对证书公钥信息计算低 96bit 的 SHA-1 摘要值，并将其作为 subjectKeyIdentifier 的值。摘要值是对证书中的主体公钥进行计算得到的，不包括标签和长度。

如果 cr 消息中包含了扩展，CA 有可能修改或者忽略其扩展。

如果 issuer 的证书和 CRL 不能够从已知的 X.500 目录中得到，则新证书中应该在 issuerAltName 中包括 URLs 信息，在 CRLDistributionPoints 控制中包括 distributionPoint 域。

如果 status 是 granted 或者 grantedWithMods，则 failInfo 域不存在。

如果 CA 拒绝用户的申请，消息体包括如下信息：

- status 是 rejected；
- failInfo 包括了合适的代码，可以是：
  - ◆ badAlg 表示 CA 不支持或者不认识 kr 指定的签名算法，
  - ◆ badPop 表示 popoSigningKey 域中的签名有误，
  - ◆ badMessageCheck 表示 PKIProtection 域中的签名有误，
  - ◆ badRequest 表示回应者不支持或者不允许该事务，
  - ◆ badTime 表示消息头中的 messageTime 的时间和回应者的系统时间相差太大，
  - ◆ badCertId 表示证书序号不能是 serial 域中所填的非零值；

如果 status 是 rejected，certifcate 域不存在。

PKIProtection 域包含了 CA 对于消息头和消息体的 DER 编码的签名。

c) 确认消息

收到 cp 的时候，证书持有者应该产生 PKIConfirm 消息。PKIHeader 中的信息和利用 RA 向 CA 申请证书的情况下的 PKIHeader 一致，除了 messageTime 是当前时间。

PKIProtection 是证书持有者使用当前有效证书的对应私钥，对于消息头和消息体的 DER 编码的签名。

6.6.9 组合证书申请

签名密钥证书和密钥管理密钥证书的申请可以由一次事务完成。RA 发起的注册请求和自我注册请求(见 6.6.2、6.6.3、6.6.4)可以和加密证书申请组合在一起(见 6.6.9)。在这样的情况下，CertReqMessages 包括了两个 CertReqMessage 的序列。一个 CertReqMessage 等同于 RA 发起的注册请求和自我注册请求的情况，另一个 CertReqMessage 等同于加密证书申请的情况。消息使用了签名证书申请的方式来加以保护。

如果组合申请中包括的是自我注册请求，则要么签名密钥证书申请成功，要么两个证书的申请都不成功。如果还需要额外的信息来提供 pop，申请者则使用自我注册请求中的私钥来对消息做签名。

签名密钥的 pop 将使用 pop 域来实现(见 6.6.2、6.6.3 和 6.6.6)。集中产生密钥对申请，私钥应该使用 6.6.8 中描述的方式进行加密。CA 的响应按照响应部分的说明。CA 响应可以是组成一个 CertRepMessage，或者是分开。申请者使用 CertReqID 来区分 CA 的响应。

响应消息的 PKIFreeText 域可以用来提供额外的信息。

收到 kp 的时候，证书持有者应该产生 PKIConfirm 消息。PKIHeader 中的信息和利用 RA 向 CA 申请证书的情况下的 PKIHeader 一致，除了 messageTime 是当前时间。

如果签名证书申请成功，PKIProtection 是证书持有者使用新的签名证书的对应私钥，对于消息头和消息体的 DER 编码的签名。如果申请被拒绝，PKIProtection 包含了对于消息头和消息体的 DER 编码的 mac，该 mac 由认证证书请求的共享秘密产生。(如果申请因为 badMessageCheck 被拒绝，则 PKIConfirm 消息不需要再产生了。)

6.6.10 从资料库请求证书

实体可以使用 LDAP V2(见 RFC 2559)向资料库请求证书。当使用 LDAP 时,实体可以通过 LDAP 搜索请求(定义见 RFC 2559)从资料库中请求证书,或是利用给定的 LDAP URL (见 RFC1959)来请求证书(即 authorityInformationAccess 扩展)。

6.6.11 从资料库请求 CRL

实体可以使用 LDAP V2(见 RFC 2559)向资料库请求 CRLs。实体可以使用 LDAP(见 RFC1777)从资料库中请求 CRLs。当使用 LDAP 时,实体可以通过 LDAP 搜索请求(定义见 RFC 2559 和 RFC1777)从资料库中请求 CRLs,或是利用给定的 LDAP URL(见 RFC1959)来请求 CRLs(即,cRLDistributionPoints 扩展中的 distributionPoint 字段)。

附 录 A
（规范性附录）
X.509 v3 证书 ASN.1

```
AuthenticationFramework {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7) 2}
DEFINITIONS ::=
BEGIN
-- EXPORTS All --
-- The types and values defined in this module are exported for use in the other ASN.1
-- modules contained within the Directory Specifications, and for the use of other applications
-- which will use them to access Directory services. Other applications may use them for
-- their own purposes, but this will not constrain extensions and modifications needed to
-- maintain or improve the Directory service.
IMPORTS
id-at, informationFramework, upperBounds selectedAttributeTypes, basicAccessControl
FROM UsefulDefinitions {joint-iso-ccitt ds(5) modules(1) usefulDefinitions(0) 2}
Name, ATTRIBUTE
FROM InformationFramework informationFramework
ub-user-password
FROM UpperBounds upperBounds
AuthenticationLevel
FROM BasicAccessControl basicAccessControl
UniqueIdentifier
FROM SelectedAttributeTypes selectedAttributeTypes ;
-- types --
Certificate ::= SIGNED {SEQUENCE{
version [0] Version DEFAULT v1,
serialNumber CertificateSerialNumber,
signature AlgorithmIdentifier,
issuer Name,
validity Validity,
subject Name,
subjectPublicKeyInfo SubjectPublicKeyInfo}
issuerUniqueIdentifier [1] IMPLICIT UniqueIdentifier OPTIONAL,
--if present, version must be v1 or v2--
subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier OPTIONAL,
--if present, version must be v1 or v2--
extensions [3] Extensions OPTIONAL
--if present, version must be v3-- }
Version ::= INTEGER {v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER
```

```
Algorithmidentifier ::= SEQUENCE{
algorithm ALGORITHM. &id({SupportedAlgorithms}),
parameters ALGORITHM. &Type ({SupportedAlgorithms}{ @algorithm}) OPTIONAL }
-- Definition of the following information object is deferred, perhaps to standardized
-- profiles of to protocol implementation conformance statements. This set is required to
-- specify a table constraint on the Parameters component of Algorithmidentifier.
-- SupportedAlgorithms ALGORITHM ::= { ... |... }
Validity ::= SEQUENCE{
notBefore ChoiceOfTime,
notAfter ChoiceOfTime }
ChoiceOfTime ::= CHOICE {
utcTime UTCTime,
generalTime GeneralizedTime }
SubjectPublicKeyInfo ::= SEQUENCE{
algorithm AlgorithmIdentifier,
subjectPublicKey BIT STRING}
Extensions ::= SEQUENCE OF Extension
Extension ::= SEQUENCE {
extnId EXTENSION. &id ({ExtensionSet}),
critical BOOLEAN DEFAULT FALSE,
extnValue OCTET STRING
-- contains a DER encoding of a value of type &ExtnType for the
-- extension object identified by extnId --
-- Definition of the following information object set is deferred, perhaps to
-- standardized profiles or to protocol implementation conformance statements.
-- The set is required to specify a table constraint on the critical component
-- of Extension.
-- ExtensionSet EXTENSION ::= { ... | ... }
EXTENSION ::= CLASS
{
&id OBJECT IDENTIFIER UNIQUE,
&ExtnType
}
WITH SYNTAX
{
SYNTAX &ExtnType
IDENTIFIED BY &id
}
Certificates ::= SEQUENCE {
certificate Certificate,
certificationPath ForwardCertificationPath OPTIONAL}
ForwardCertificationPath ::= SEQUENCE OF CrossCertificates
CertificationPath ::= SEQUENCE {
```

```
userCertificate Certificate,
theCACertificates SEQUENCE OF CertificatePair OPTIONAL}
CrossCertificates ::= SET OF Certificate
CertificateList ::= SIGNED { SEQUENCE {
version Version OPTIONAL, -- if present, must be v2
signature AlgorithmIdentifier,
issuer Name,
thisUpdate ChoiceOfTime,
nextUpdate ChoiceOfTime OPTIONAL,
revokedCertificates SEQUENCE OF SEQUENCE {
userCertificate CertificateSerialNumber,
revocationDate ChoiceOfTime,
crlEntryExtensions Extensions OPTIONAL } OPTIONAL,
crlExtensions [0] Extensions OPTIONAL }}
CertificatePair ::= SEQUENCE {
forward [0] Certificate OPTIONAL,
reverse [1] Certificate OPTIONAL
-- at least one of the pair shall be present -- }
-- attribute types--
userPassword ATTRIBUTE ::= {
WITH SYNTAX OCTET STRING (SIZE (0..ub-user-password))
EQUALITY MATCHING RULE octetStringMatch
ID id-at-userPassword }
userCertificate ATTRIBUTE ::= {
WITH SYNTAX Certificate
ID id-at-userCertificate }
cACertificate ATTRIBUTE ::= {
WITH SYNTAX Certificate
ID id-at-cACertificate }
authorityRevocationList ATTRIBUTE ::= {
WITH SYNTAX CertificateList
ID id-at-authorityRevocationList }
certificateRevocationList ATTRIBUTE ::= {
WITH SYNTAX CertificateList
ID id-at-certificateRevocationList }
crossCertificatePair ATTRIBUTE ::= {
WITH SYNTAX CertificatePair
ID id-at-crossCertificatePair }
-- information object classes --
ALGORITHM ::= TYPE-IDENTIFIER
-- Parameterized Types --
HASHED {ToBeHashed} ::= OCTET STRING ( CONSTRAINED-BY {
--must be the result of applying a hashing procedure to the --
```

```
--DER-encoded octets of a value of -- ToBeHashed }}
ENCRYPTED { To/BeEnciphered} ::= BIT STRING ( CONSTRAINED BY {
--must be the result of applying an encipherment procedure to the --
--BER-encoded octets of a value of -- ToBeEnciphered }}
SIGNED { ToBeSigned } ::= SEQUENCE{
ToBeSigned,
COMPONENTS OF SIGNATURE { ToBeSigned }},
SIGNATURE { OfSignature } ::= SEQUENCE {
AlgorithmIdentifier,
ENCRYPTED { HASHED { OfSignature }}}
-- object identifier assignments --
id-at-userPassword OBJECT IDENTIFIER ::= {id-at 35}
id-at-userCertificate OBJECT IDENTIFIER ::= {id-at 36}
id-at-cAcertificate OBJECT IDENTIFIER ::= {id-at 37}
id-at-authorityRevocationList OBJECT IDENTIFIER ::= {id-at 38}
id-at-certificateRevocationList OBJECT IDENTIFIER ::= {id-at 39}
id-at-crossCertificatePair OBJECT IDENTIFIER ::= {id-at 40}
id-at-supportedAlgorithms OBJECT IDENTIFIER ::= {id-at 52}
id-at-deltaRevocationList OBJECT IDENTIFIER ::= {id-at 53}
END
```

附 录 B
（规范性附录）
证书和 CRL 扩展 ASN.1

```
CertificateExtensions {joint-iso-ccitt ds(5) module(1) certificateExtensions(26) 0}
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
-- EXPORTS ALL --
IMPORTS
id-at, id-ce, id-mr, informationFramework, authenticationFramework,
selectedAttributeTypes, upperBounds
FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1)
usefulDefinitions(0) 2}
Name, RelativeDistinguishedName, ATTRIBUTE, Attribute,
MATCHING-RULE FROM InformationFramework informationFramework
CertificateSerialNumber, CertificateList, AlgorithmIdentifier,
EXTENSION
FROM AuthenticationFramework authenticationFramework
DirectoryString
FROM SelectedAttributeTypes selectedAttributeTypes
ub-name
FROM UpperBounds upperBounds
ORAddress
FROM MTSAbstractService {joint-iso-ccitt mhs(6) mts(3)
modules(0) mts-abstract-service(1) version—1994 (0) } ;
-- Unless explicitly noted otherwise, there is no significance to the ordering
-- of components of a SEQUENCE OF construct in this specification.
-- Key and policy information extensions --
authorityKeyIdentifier EXTENSION ::= {
SYNTAX AuthorityKeyIdentifier
IDENTIFIED BY { id-ce 35 } }
AuthorityKeyIdentifier ::= SEQUENCE {
keyIdentifier [0] KeyIdentifier OPTIONAL,
authorityCertIssuer [1] GeneralNames OPTIONAL,
authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
( WITH COMPONENTS {..., authorityCertIssuer PRESENT,
authorityCertSerialNumber PRESENT} |
WITH COMPONENTS {..., authorityCertIssuer ABSENT,
authorityCertSerialNumber ABSENT} )
KeyIdentifier ::= OCTET STRING
subjectKeyIdentifier EXTENSION ::= {
```

```
SYNTAX SubjectKeyIdentifier
IDENTIFIED BY { id-ce 14 } }
SubjectKeyIdentifier ::= KeyIdentifier
keyUsage EXTENSION ::= {
SYNTAX KeyUsage
IDENTIFIED BY { id-ce 15 } }
KeyUsage ::= BIT STRING {
digitalSignature (0),
nonRepudiation (1),
keyEncipherment (2),
dataEncipherment (3),
keyAgreement (4),
keyCertSign (5),
cRLSign (6) }
privateKeyUsagePeriod EXTENSION ::= {
SYNTAX PrivateKeyUsagePeriod
IDENTIFIED BY { id-ce 16 } }
PrivateKeyUsagePeriod ::= SEQUENCE {
notBefore [0] GeneralizedTime OPTIONAL,
notAfter [1] GeneralizedTime OPTIONAL }
( WITH COMPONENTS {..., notBefore PRESENT} |
WITH COMPONENTS {..., notAfter PRESENT} )
certificatePolicies EXTENSION ::= {
SYNTAX CertificatePoliciesSyntax
IDENTIFIED BY { id-ce 32 } }
CertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation
PolicyInformation ::= SEQUENCE {
policyIdentifier CertPolicyId,
policyQualifiers SEQUENCE SIZE (1..MAX) OF
PolicyQualifierInfo OPTIONAL }
CertPolicyId ::= OBJECT IDENTIFIER
PolicyQualifierInfo ::= SEQUENCE {
policyQualifierId CERT-POLICY-QUALIFIER.&id
({SupportedPolicyQualifiers}),
qualifier CERT-POLICY-QUALIFIER.&Qualifier
({SupportedPolicyQualifiers}{@policyQualifierId})
OPTIONAL }
SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { ... }
CERT-POLICY-QUALIFIER ::= CLASS {
&id OBJECT IDENTIFIER UNIQUE,
&Qualifier OPTIONAL }
WITH SYNTAX {
POLICY-QUALIFIER-ID &id
```

```
[QUALIFIER-TYPE &Qualifier] }
policyMappings EXTENSION ::= {
SYNTAX PolicyMappingsSyntax
IDENTIFIED BY { id-ce 33 } }
PolicyMappingsSyntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
issuerDomainPolicy CertPolicyId,
subjectDomainPolicy CertPolicyId }
supportedAlgorithms ATTRIBUTE ::= {
WITH SYNTAX SupportedAlgorithm
EQUALITY MATCHING RULE algorithmIdentifierMatch
ID { id-at 52 } }
SupportedAlgorithm ::= SEQUENCE {
algorithmIdentifier AlgorithmIdentifier,
intendedUsage [0] KeyUsage OPTIONAL,
intendedCertificatePolicies [1] CertificatePoliciesSyntax OPTIONAL }
-- Certificate subject and certificate issuer attributes extensions --
subjectAltName EXTENSION ::= {
SYNTAX GeneralNames
IDENTIFIED BY { id-ce 17 } }
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
GeneralName ::= CHOICE {
otherName [0] INSTANCE OF OTHER-NAME,
RFC 822Name [1] IA5String,
dNSName [2] IA5String,
x400Address [3] ORAddress,
directoryName [4] Name,
ediPartyName [5] EDIPartyName,
uniformResourceIdentifier [6] IA5String,
iPAddress [7] OCTET STRING,
registeredID [8] OBJECT IDENTIFIER }
OTHER-NAME ::= TYPE-IDENTIFIER
EDIPartyName ::= SEQUENCE {
nameAssigner [0] DirectoryString {ub-name} OPTIONAL,
partyName [1] DirectoryString {ub-name} }
issuerAltName EXTENSION ::= {
SYNTAX GeneralNames
IDENTIFIED BY { id-ce 18 } }
subjectDirectoryAttributes EXTENSION ::= {
SYNTAX AttributesSyntax
IDENTIFIED BY { id-ce 9 } }
AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
-- Certification path constraints extensions --
basicConstraints EXTENSION ::= {
```

```
SYNTAX BasicConstraintsSyntax
IDENTIFIED BY { id-ce 19 } }
BasicConstraintsSyntax ::= SEQUENCE {
cA BOOLEAN DEFAULT FALSE,
pathLenConstraint INTEGER (0..MAX) OPTIONAL }
nameConstraints EXTENSION ::= {
SYNTAX NameConstraintsSyntax
IDENTIFIED BY { id-ce 30 } }
NameConstraintsSyntax ::= SEQUENCE {
permittedSubtrees [0] GeneralSubtrees OPTIONAL,
excludedSubtrees [1] GeneralSubtrees OPTIONAL }
GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
GeneralSubtree ::= SEQUENCE {
base GeneralName,
minimum [0] BaseDistance DEFAULT 0,
maximum [1] BaseDistance OPTIONAL }
BaseDistance ::= INTEGER (0..MAX)
policyConstraints EXTENSION ::= {
SYNTAX PolicyConstraintsSyntax
IDENTIFIED BY { id-ce 36 } }
PolicyConstraints Syntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
requireExplicitPolicy [0] SkipCerts OPTIONAL,
inhibitPolicyMapping [1] SkipCerts OPTIONAL }
SkipCerts ::= INTEGER (0..MAX)
-- Basic CRL extensions --
cRLNumber EXTENSION ::= {
SYNTAX CRLNumber
IDENTIFIED BY { id-ce 20 } }
CRLNumber ::= INTEGER (0..MAX)
reasonCode EXTENSION ::= {
SYNTAX CRLReason
IDENTIFIED BY { id-ce 21 } }
CRLReason ::= ENUMERATED {
unspecified (0),
keyCompromise (1),
cACompromise (2),
affiliationChanged (3),
superseded (4),
cessationOfOperation (5),
certificateHold (6),
removeFromCRL (8) }
instructionCode EXTENSION ::= {
SYNTAX HoldInstruction
```

```
IDENTIFIED BY { id-ce 23 } }
HoldInstruction ::= OBJECT IDENTIFIER
invalidityDate EXTENSION ::= {
SYNTAX GeneralizedTime
IDENTIFIED BY { id-ce 24 } }
-- CRL distribution points and delta-CRL extensions --
cRLDistributionPoints EXTENSION ::= {
SYNTAX CRLDistPointsSyntax
IDENTIFIED BY { id-ce 31 } }
CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
distributionPoint [0] DistributionPointName OPTIONAL,
reasons [1] ReasonFlags OPTIONAL,
cRLIssuer [2] GeneralNames OPTIONAL }
DistributionPointName ::= CHOICE {
fullName [0] GeneralNames,
nameRelativeToCRLIssuer [1] RelativeDistinguishedName }
ReasonFlags ::= BIT STRING {
unused (0),
keyCompromise (1),
caCompromise (2),
affiliationChanged (3),
superseded (4),
cessationOfOperation (5),
certificateHold (6) }
issuingDistributionPoint EXTENSION ::= {
SYNTAX IssuingDistPointSyntax
IDENTIFIED BY { id-ce 28 } }
IssuingDistPointSyntax ::= SEQUENCE {
distributionPoint [0] DistributionPointName OPTIONAL,
onlyContainsUserCerts [1] BOOLEAN DEFAULT FALSE,
onlyContainsCACerts [2] BOOLEAN DEFAULT FALSE,
onlySomeReasons [3] ReasonFlags OPTIONAL,
indirectCRL [4] BOOLEAN DEFAULT FALSE }
certificateIssuer EXTENSION ::= {
SYNTAX GeneralNames
IDENTIFIED BY { id-ce 29 } }
deltaCRLIndicator EXTENSION ::= {
SYNTAX BaseCRLNumber
IDENTIFIED BY { id-ce 27 } }
BaseCRLNumber ::= CRLNumber
deltaRevocationList ATTRIBUTE ::= {
WITH SYNTAX CertificateList
```

```
EQUALITY MATCHING RULE certificateListExactMatch
ID {id-at 53 } }
-- Matching rules --
certificateExactMatch MATCHING-RULE ::= {
SYNTAX CertificateExactAssertion
ID id-mr-certificateExactMatch }
CertificateExactAssertion ::= SEQUENCE {
serialNumber CertificateSerialNumber,
issuer Name }
certificateMatch MATCHING-RULE ::= {
SYNTAX CertificateAssertion
ID id-mr-certificateMatch }
CertificateAssertion ::= SEQUENCE {
serialNumber [0] CertificateSerialNumber OPTIONAL,
issuer [1] Name OPTIONAL,
subjectKeyIdentifier [2] SubjectKeyIdentifier OPTIONAL,
authorityKeyIdentifier [3] AuthorityKeyIdentifier OPTIONAL,
certificateValid [4] UTCTime OPTIONAL,
privateKeyValid [5] GeneralizedTime OPTIONAL,
subjectPublicKeyAlgID [6] OBJECT IDENTIFIER OPTIONAL,
keyUsage [7] KeyUsage OPTIONAL,
subjectAltName [8] AltNameType OPTIONAL,
policy [9] CertPolicySet OPTIONAL,
pathToName [10] Name OPTIONAL }
AltNameType ::= CHOICE {
builtinNameForm ENUMERATED {
RFC 822Name (1),
dNSName (2),
x400Address (3),
directoryName (4),
ediPartyName (5),
uniformResourceIdentifier (6),
iPAddress (7),
registeredId (8) },
otherNameForm OBJECT IDENTIFIER }
certificatePairExactMatch MATCHING-RULE ::= {
SYNTAX CertificatePairExactAssertion
ID id-mr-certificatePairExactMatch }
CertificatePairExactAssertion ::= SEQUENCE {
forwardAssertion [0] CertificateExactAssertion OPTIONAL,
reverseAssertion [1] CertificateExactAssertion OPTIONAL }
( WITH COMPONENTS {..., forwardAssertion PRESENT} |
WITH COMPONENTS {..., reverseAssertion PRESENT} )
```

```
certificatePairMatch MATCHING-RULE ::= {
SYNTAX CertificatePairAssertion
ID id-mr-certificatePairMatch }
CertificatePairAssertion ::= SEQUENCE {
forwardAssertion [0] CertificateAssertion OPTIONAL,
reverseAssertion [1] CertificateAssertion OPTIONAL }
( WITH COMPONENTS {..., forwardAssertion PRESENT} |
WITH COMPONENTS {..., reverseAssertion PRESENT} )
certificateListExactMatch MATCHING-RULE ::= {
SYNTAX CertificateListExactAssertion
ID id-mr-certificateListExactMatch }
CertificateListExactAssertion ::= SEQUENCE {
issuer Name,
thisUpdate UTCTime,
distributionPoint DistributionPointName OPTIONAL }
certificateListMatch MATCHING-RULE ::= {
SYNTAX CertificateListAssertion
ID id-mr-certificateListMatch }
CertificateListAssertion ::= SEQUENCE {
issuer Name OPTIONAL,
minCRLNumber [0] CRLNumber OPTIONAL,
maxCRLNumber [1] CRLNumber OPTIONAL,
reasonFlags ReasonFlags OPTIONAL,
dateAndTime UTCTime OPTIONAL,
distributionPoint [2] DistributionPointName OPTIONAL }
algorithmIdentifierMatch MATCHING-RULE ::= {
SYNTAX AlgorithmIdentifier
ID id-mr-algorithmIdentifierMatch }
-- Object identifier assignments --
id-at-supportedAlgorithms OBJECT IDENTIFIER ::= {id-at 52}
id-at-deltaRevocationList OBJECT IDENTIFIER ::= {id-at 53}
id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::= {id-ce 9}
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::= {id-ce 14}
id-ce-keyUsage OBJECT IDENTIFIER ::= {id-ce 15}
id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::= {id-ce 16}
id-ce-subjectAltName OBJECT IDENTIFIER ::= {id-ce 17}
id-ce-issuerAltName OBJECT IDENTIFIER ::= {id-ce 18}
id-ce-basicConstraints OBJECT IDENTIFIER ::= {id-ce 19}
id-ce-cRLNumber OBJECT IDENTIFIER ::= {id-ce 20}
id-ce-reasonCode OBJECT IDENTIFIER ::= {id-ce 21}
id-ce-instructionCode OBJECT IDENTIFIER ::= {id-ce 23}
id-ce-invalidityDate OBJECT IDENTIFIER ::= {id-ce 24}
id-ce-deltaCRLIndicator OBJECT IDENTIFIER ::= {id-ce 27}
```

id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= {id-ce 28}
id-ce-certificateIssuer OBJECT IDENTIFIER ::= {id-ce 29}
id-ce-nameConstraints OBJECT IDENTIFIER ::= {id-ce 30}
id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::= {id-ce 31}
id-ce-certificatePolicies OBJECT IDENTIFIER ::= {id-ce 32}
id-ce-policyMappings OBJECT IDENTIFIER ::= {id-ce 33}
id-ce-policyConstraints OBJECT IDENTIFIER ::= {id-ce 34}
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= {id-ce 35}
id-mr-certificateExactMatch OBJECT IDENTIFIER ::= {id-mr 34}
id-mr-certificateMatch OBJECT IDENTIFIER ::= {id-mr 35}
id-mr-certificatePairExactMatch OBJECT IDENTIFIER ::= {id-mr 36}
id-mr-certificatePairMatch OBJECT IDENTIFIER ::= {id-mr 37}
id-mr-certificateListExactMatch OBJECT IDENTIFIER ::= {id-mr 38}
id-mr-certificateListMatch OBJECT IDENTIFIER ::= {id-mr 39}
id-mr-algorithmIdentifierMatch OBJECT IDENTIFIER ::= {id-mr 40}
– The following OBJECT IDENTIFIERS are not used by this specification:
– {id-ce 2}, {id-ce 3}, {id-ce 4}, {id-ce 5}, {id-ce 6}, {id-ce 7},
– {id-ce 8}, {id-ce 10}, {id-ce 11}, {id-ce 12}, {id-ce 13},
– {id-ce 22}, {id-ce 25}, {id-ce 26}
END

附 录 C
（规范性附录）
ASN.1 Module for transactions

The following section contains the complete ASN.1 module from RFC 2510, the Certificate Management Protocol. Only a small subset of the messages defined in RFC2510 are required to implement this specification. The entire module is provided for completeness. Information about messages defined by this ASN.1 module but not used in the MISPC may be found in RFC2510.

```
PKIX-CMP DEFINITIONS ::=
BEGIN -- EXPLICT TAGS
IMPORTS
CertReqMessages, EncryptedValue, EncryptedKey
FROM CMRF
PKIMessage ::= SEQUENCE {
Header PKIHeader,
Body PKIBody,
Protection [0] PKIProtection OPTIONAL,
ExtraCerts [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL
}
PKIHeader ::= SEQUENCE {
pvno INTEGER { ietf-version2 (1) },
sender GeneralName,
-- identifies the sender
recipient GeneralName,
-- identifies the intended recipient
messageTime [0] GeneralizedTime OPTIONAL,
-- time of production of this message (used when sender
-- believes that the transport will be "suitable"; i.e.,
-- that the time will still be meaningful upon receipt)
protectionAlg [1] AlgorithmIdentifier OPTIONAL,
-- algorithm used for calculation of protection bits
senderKID [2] KeyIdentifier OPTIONAL,
recipKID [3] KeyIdentifier OPTIONAL,
-- to identify specific keys used for protection
transactionID [4] OCTET STRING OPTIONAL,
-- identifies the transaction; i.e., this will be the same in
-- corresponding request, response and confirmation messages
senderNonce [5] OCTET STRING OPTIONAL,
recipNonce [6] OCTET STRING OPTIONAL,
-- nonces used to provide replay protection, senderNonce
```

```
-- is inserted by the creator of this message; recipNonce
-- is a nonce previously inserted in a related message by
-- the intended recipient of this message
freeText [7] PKIFreeText OPTIONAL,
-- this may be used to indicate context-specific instructions
-- (this field is intended for human consumption)
generalInfo [8] SEQUENCE SIZE (1..MAX) OF
InfoTypeAndValue OPTIONAL
-- this may be used to convey context-specific information
-- (this field not primarily intended for human consumption)
}
PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
-- text encoded as UTF-8 String (note: each UTF8String SHOULD
-- include an RFC 1766 language tag to indicate the language
-- of the contained text)
PKIBody ::= CHOICE { -- message-specific body elements
ir [0] CertReqMessages, --Initialization Request
ip [1] CertRepMessage, --Initialization Response
cr [2] CertReqMessages, --Certification Request
cp [3] CertRepMessage, --Certification Response
p10cr [4] CertificationRequest, --imported from [PKCS10]
popdecc [5] POPODecKeyChallContent, --pop Challenge
popdecr [6] POPODecKeyRespContent, --pop Response
kur [7] CertReqMessages, --Key Update Request
kup [8] CertRepMessage, --Key Update Response
krr [9] CertReqMessages, --Key Recovery Request
krp [10] KeyRecRepContent, --Key Recovery Response
rr [11] RevReqContent, --Revocation Request
rp [12] RevRepContent, --Revocation Response
ccr [13] CertReqMessages, --Cross-Cert. Request
ccp [14] CertRepMessage, --Cross-Cert. Response
ckuann [15] CAKeyUpdAnnContent, --CA Key Update Ann.
cann [16] CertAnnContent, --Certificate Ann.
rann [17] RevAnnContent, --Revocation Ann.
crlann [18] CRLAnnContent, --CRL Announcement
conf [19] PKIConfirmContent, --Confirmation
nested [20] NestedMessageContent, --Nested Message
genm [21] GenMsgContent, --General Message
genp [22] GenRepContent, --General Response
error [23] ErrorMsgContent --Error Message
}
PKIProtection ::= BIT STRING
ProtectedPart ::= SEQUENCE {
```

```
header PKIHeader,
body PKIBody
}
PasswordBasedMac ::= OBJECT IDENTIFIER --{1 2 840 113533 7 66 13}
PBMParameter ::= SEQUENCE {
salt OCTET STRING,
owf AlgorithmIdentifier,
-- AlgId for a One-Way Function (SHA-1 recommended)
iterationCount INTEGER,
-- number of times the OWF is applied
mac AlgorithmIdentifier
-- the mac AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
} -- or HMAC [RFC2104, RFC2202])
DHBasedMac ::= OBJECT IDENTIFIER --{1 2 840 113533 7 66 30}
DHBMParameter ::= SEQUENCE {
owf AlgorithmIdentifier,
-- AlgId for a One-Way Function (SHA-1 recommended)
mac AlgorithmIdentifier
-- the MAC AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
} -- or HMAC [RFC2104, RFC2202])
NestedMessageContent ::= PKIMessage
PKIStatus ::= INTEGER {
granted (0),
-- you got exactly what you asked for
grantedWithMods (1),
-- you got something like what you asked for; the
-- requester is responsible for ascertaining the differences
rejection (2),
-- you don't get it, more information elsewhere in the message
waiting (3),
-- the request body part has not yet been processed,
-- expect to hear more later
revocationWarning (4),
-- this message contains a warning that a revocation is
-- imminent
revocationNotification (5),
-- notification that a revocation has occurred
keyUpdateWarning (6)
-- update already done for the oldCertId specified in
-- CertReqMsg
}
PKIFailureInfo ::= BIT STRING {
-- since we can fail in more than one way!
```

```
-- More codes may be added in the future if/when required.
badAlg (0),
-- unrecognized or unsupported Algorithm Identifier
badMessageCheck (1),
-- integrity check failed (e.g., signature did not verify)
badRequest (2),
-- transaction not permitted or supported
badTime (3),
-- messageTime was not sufficiently close to the system time,
-- as defined by local policy
badCertId (4),
-- no certificate could be found matching the provided criteria
badDataFormat (5),
-- the data submitted has the wrong format
wrongAuthority (6),
-- the authority indicated in the request is different from the
-- one creating the response token
incorrectData (7),
-- the requester's data is incorrect (for notary services)
missingTimeStamp (8),
-- when the timestamp is missing but should be there (by policy)
badPoP (9)
-- when proof of possession does not verify
}
PKIStatusInfo ::= SEQUENCE {
status PKIStatus,
statusString PKIFreeText OPTIONAL,
failInfo PKIFailureInfo OPTIONAL
}
OOBCert ::= Certificate
OOBCertHash ::= SEQUENCE {
hashAlg [0] AlgorithmIdentifier OPTIONAL,
certId [1] CertId OPTIONAL,
hashVal BIT STRING
-- hashVal is calculated over DER encoding of the
-- subjectPublicKey field of the corresponding cert.
}
POPODecKeyChallContent ::= SEQUENCE OF Challenge
-- One Challenge per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages).
Challenge ::= SEQUENCE {
owf AlgorithmIdentifier OPTIONAL,
-- MUST be present in the first Challenge; MAY be omitted in any
```

```
-- subsequent Challenge in POPODecKeyChallContent (if omitted,
-- then the owf used in the immediately preceding Challenge is
-- to be used).
witness OCTET STRING,
-- the result of applying the one-way function (owf) to a
-- randomly-generated INTEGER, A. [Note that a different
-- INTEGER MUST be used for each Challenge.]
challenge OCTET STRING
-- the encryption (under the public key for which the cert.
-- request is being made) of Rand, where Rand is specified as
-- Rand ::= SEQUENCE {
-- int INTEGER,
--  - the randomly-generated INTEGER A (above)
-- sender GeneralName
--  - the sender's name (as included in PKIHeader)
-- }
}
POPODecKeyRespContent ::= SEQUENCE OF INTEGER
-- One INTEGER per encryption key certification request (in the
-- same order as these requests appear in CertReqMessages). The
-- retrieved INTEGER A (above) is returned to the sender of the
-- corresponding Challenge.
CertRepMessage ::= SEQUENCE {
caPubs [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL,
response SEQUENCE OF CertResponse
}
CertResponse ::= SEQUENCE {
certReqId INTEGER,
-- to match this response with corresponding request (a value
-- of -1 is to be used if certReqId is not specified in the
-- corresponding request)
status PKIStatusInfo,
certifiedKeyPair CertifiedKeyPair OPTIONAL,
rspInfo OCTET STRING OPTIONAL
-- analogous to the id-regInfo-asciiPairs OCTET STRING defined
-- for regInfo in CertReqMsg [CRMF]
}
CertifiedKeyPair ::= SEQUENCE {
certOrEncCert CertOrEncCert,
privateKey [0] EncryptedValue OPTIONAL,
publicationInfo [1] PKIPublicationInfo OPTIONAL
}
CertOrEncCert ::= CHOICE {
```

```
certificate [0] Certificate,
encryptedCert [1] EncryptedValue
}
KeyRecRepContent ::= SEQUENCE {
status PKIStatusInfo,
newSigCert [0] Certificate OPTIONAL,
caCerts [1] SEQUENCE SIZE (1..MAX) OF
Certificate OPTIONAL,
keyPairHist [2] SEQUENCE SIZE (1..MAX) OF
CertifiedKeyPair OPTIONAL
}
RevReqContent ::= SEQUENCE OF RevDetails
RevDetails ::= SEQUENCE {
certDetails CertTemplate,
-- allows requester to specify as much as they can about
-- the cert. for which revocation is requested
-- (e.g., for cases in which serialNumber is not available)
revocationReason ReasonFlags OPTIONAL,
-- the reason that revocation is requested
badSinceDate GeneralizedTime OPTIONAL,
-- indicates best knowledge of sender
crlEntryDetails Extensions OPTIONAL
-- requested crlEntryExtensions
}
RevRepContent ::= SEQUENCE {
status SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
-- in same order as was sent in RevReqContent
revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL,
-- IDs for which revocation was requested (same order as status)
crls [1] SEQUENCE SIZE (1..MAX) OF CertificateList OPTIONAL
-- the resulting CRLs (there may be more than one)
}
CAKeyUpdAnnContent ::= SEQUENCE {
oldWithNew Certificate, -- old pub signed with new priv
newWithOld Certificate, -- new pub signed with old priv
newWithNew Certificate -- new pub signed with new priv
}
CertAnnContent ::= Certificate
RevAnnContent ::= SEQUENCE {
status PKIStatus,
certId CertId,
willBeRevokedAt GeneralizedTime,
badSinceDate GeneralizedTime,
```

```
crlDetails Extensions OPTIONAL
-- extra CRL details(e.g., crl number, reason, location, etc.)
}
CRLAnnContent ::= SEQUENCE OF CertificateList
PKIConfirmContent ::= NULL
InfoTypeAndValue ::= SEQUENCE {
infoType OBJECT IDENTIFIER,
infoValue ANY DEFINED BY infoType OPTIONAL
}
-- Example InfoTypeAndValue contents include, but are not limited to:
--  { CAProtEncCert = {id-it 1}, Certificate }
--  { SignKeyPairTypes = {id-it 2}, SEQUENCE OF AlgorithmIdentifier }
--  { EncKeyPairTypes = {id-it 3}, SEQUENCE OF AlgorithmIdentifier }
--  { PreferredSymmAlg = {id-it 4}, AlgorithmIdentifier }
--  { CAKeyUpdateInfo = {id-it 5}, CAKeyUpdAnnContent }
--  { CurrentCRL = {id-it 6}, CertificateList }
-- where {id-it} = {id-pkix 4} = {1 3 6 1 5 5 7 4}
-- This construct MAY also be used to define new PKIX Certificate
-- Management Protocol request and response messages, or general-
-- purpose (e.g., announcement) messages for future needs or for
-- specific environments.
GenMsgContent ::= SEQUENCE OF InfoTypeAndValue
-- May be sent by EE, RA, or CA (depending on message content).
-- The OPTIONAL infoValue parameter of InfoTypeAndValue will typically
-- be omitted for some of the examples given above. The receiver is
-- free to ignore any contained OBJ. IDs that it does not recognize.
-- If sent from EE to CA, the empty set indicates that the CA may send
-- any/all information that it wishes.
GenRepContent ::= SEQUENCE OF InfoTypeAndValue
-- The receiver is free to ignore any contained OBJ. IDs that it does
-- not recognize.
ErrorMsgContent ::= SEQUENCE {
pKIStatusInfo PKIStatusInfo,
errorCode INTEGER OPTIONAL,
-- implementation-specific error codes
errorDetails PKIFreeText OPTIONAL
-- implementation-specific error details
}
```

附 录 D
（规范性附录）
证书请求消息格式 ASN.1 Module

The following section contains the complete ASN.1 module from RFC 2511, the Certificate Request Message Format. Only a small subset of the structures defined in RFC2511 are required to implement this specification. The entire module is provided for completeness. Information about structures defined by this ASN.1 module but not used in the MISPC may be found in RFC2511.

```
CRMF DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
-- Directory Authentication Framework (X.509)
Version, AlgorithmIdentifier, Name, Time,
SubjectPublicKeyInfo, Extensions, UniqueIdentifier
FROM AuthenticationFramework { joint-iso-itu-t ds(5)
module(1) authenticationFramework(7) 3 }
-- Certificate Extensions (X.509)
GeneralName
FROM CertificateExtensions {joint-iso-ccitt ds(5)
module(1) certificateExtensions(26) 0}
-- Cryptographic Message Syntax
EnvelopedData
FROM CryptographicMessageSyntax { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
modules(0) cms(1) };
CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg
CertReqMsg ::= SEQUENCE {
CertReq CertRequest,
Pop ProofOfPossession OPTIONAL,
-- content depends upon key type
regInfo SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue OPTIONAL }
CertRequest ::= SEQUENCE {
CertReqId INTEGER, -- ID for matching request and reply
CertTemplate CertTemplate, -- Selected fields of cert to be issued
Controls Controls OPTIONAL } -- Attributes affecting issuance
CertTemplate ::= SEQUENCE {
Version [0] Version OPTIONAL,
serialNumber [1] INTEGER OPTIONAL,
signingAlg [2] AlgorithmIdentifier OPTIONAL,
issuer [3] Name OPTIONAL,
```

```
validity [4] OptionalValidity OPTIONAL,
subject [5] Name OPTIONAL,
publicKey [6] SubjectPublicKeyInfo OPTIONAL,
issuerUID [7] UniqueIdentifier OPTIONAL,
subjectUID [8] UniqueIdentifier OPTIONAL,
extensions [9] Extensions OPTIONAL }
OptionalValidity ::= SEQUENCE {
notBefore [0] Time OPTIONAL,
notAfter [1] Time OPTIONAL } --at least one MUST be present
Controls ::= SEQUENCE SIZE(1..MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
type OBJECT IDENTIFIER,
value ANY DEFINED BY type }
ProofOfPossession ::= CHOICE {
raVerified [0] NULL,
-- used if the RA has already verified that the requester is in
-- possession of the private key
signature [1] POPOSigningKey,
keyEncipherment [2] POPOPrivKey,
keyAgreement [3] POPOPrivKey }
POPOSigningKey ::= SEQUENCE {
poposkInput [0] POPOSKInput OPTIONAL,
algorithmIdentifier AlgorithmIdentifier,
signature BIT STRING }
-- The signature (using "algorithmIdentifier") is on the
-- DER-encoded value of popInput. NOTE: If poposkInput is present
-- in the pop field, popInput is constructed
-- with otherinput. If poposkInput is not present, subject is the name
-- from CertTemplate. Note that the encoding of PopInput is
intentionally ambiguous.
PoposkInput ::= CHOICE {
Subject name,
Sender [0] generalName,
publicKeyMAC [1] PKMACValue
}
-- The pop is calculated upon the structure popInput, which is defined
-- as follows:
PopInput ::= SEQUENCE {
CHOICE {
otherinput popskInput,
subject name },
publicKey subjectpublicKey
}
```

```
-- If poposkInput is present
-- in the pop field, popInput is constructed
-- with otherinput. If poposkInput is not present, subject is the name
-- from CertTemplate. Note that the encoding of PopInput is
-- intentionally ambiguous.
PKMACValue ::= SEQUENCE {
algId AlgorithmIdentifier,
-- algorithm value shall be PasswordBasedMac {1 2 840 113533 7 66 13}
-- parameter value is PBMParameter
value BIT STRING }
PBMParameter ::= SEQUENCE {
salt OCTET STRING,
owf AlgorithmIdentifier,
-- AlgId for a One-Way Function (SHA-1 recommended)
iterationCount INTEGER,
-- number of times the OWF is applied
mac AlgorithmIdentifier
-- the MAC AlgId (e.g., DES-MAC, Triple-DES-MAC [PKCS11],
} -- or HMAC [RFC2104, RFC2202])
POPOPrivKey ::= CHOICE {
thisMessage [0] BIT STRING,
-- posession is proven in this message (which contains the private
-- key itself (encrypted for the CA))
subsequentMessage [1] SubsequentMessage,
-- possession will be proven in a subsequent message
dhMAC [2] BIT STRING }
-- for keyAgreement (only), possession is proven in this message
-- (which contains a MAC (over the DER-encoded value of the
-- certReq parameter in CertReqMsg, which MUST include both subject
-- and publicKey) based on a key derived from the end entity's
-- private DH key and the CA's public DH key);
-- the dhMAC value MUST be calculated as per the directions given
-- in Appendix A.
SubsequentMessage ::= INTEGER {
encrCert (0),
-- requests that resulting certificate be encrypted for the
-- end entity (following which, POP will be proven in a
-- confirmation message)
challengeResp (1) }
-- requests that CA engage in challenge-response exchange with
-- end entity in order to prove private key possession
-- Object identifier assignments --
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
```

```
dod(6) internet(1) security(5) mechanisms(5) 7 }
-- arc for Internet X.509 PKI protocols and their components
id-pkip OBJECT IDENTIFIER ::= { id-pkix 5 }
-- Registration Controls in CRMF
id-regCtrl OBJECT IDENTIFIER ::= { id-pkip 1 }
-- The following definition may be uncommented for use with
-- ASN.1 compilers which do not understand UTF8String.
-- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING
id-regCtrl-regToken OBJECT IDENTIFIER ::= { id-regCtrl 1 }
--with syntax:
RegToken ::= UTF8String
id-regCtrl-authenticator OBJECT IDENTIFIER ::= { id-regCtrl 2 }
--with syntax:
Authenticator ::= UTF8String
id-regCtrl-pkiPublicationInfo OBJECT IDENTIFIER ::= { id-regCtrl 3 }
--with syntax:
PKIPublicationInfo ::= SEQUENCE {
action INTEGER {
dontPublish (0),
pleasePublish (1) },
pubInfos SEQUENCE SIZE (1..MAX) OF SinglePubInfo OPTIONAL }
-- pubInfos MUST NOT be present if action is "dontPublish"
-- (if action is "pleasePublish" and pubInfos is omitted,
-- "dontCare" is assumed)
SinglePubInfo ::= SEQUENCE {
pubMethod INTEGER {
dontCare (0),
x500 (1),
web (2),
ldap (3) },
pubLocation GeneralName OPTIONAL }
id-regCtrl-pkiArchiveOptions OBJECT IDENTIFIER ::= { id-regCtrl 4 }
--with syntax:
PKIArchiveOptions ::= CHOICE {
encryptedPrivKey [0] EncryptedKey,
-- the actual value of the private key
keyGenParameters [1] KeyGenParameters,
-- parameters which allow the private key to be re-generated
archiveRemGenPrivKey [2] BOOLEAN }
-- set to TRUE if sender wishes receiver to archive the private
-- key of a key pair which the receiver generates in response to
-- this request; set to FALSE if no archival is desired.
EncryptedKey ::= CHOICE {
```

```
encryptedValue EncryptedValue,
envelopedData [0] EnvelopedData }
-- The encrypted private key MUST be placed in the envelopedData
-- encryptedContentInfo encryptedContent OCTET STRING.
EncryptedValue ::= SEQUENCE {
intendedAlg [0] AlgorithmIdentifier OPTIONAL,
-- the intended algorithm for which the value will be used
symmAlg [1] AlgorithmIdentifier OPTIONAL,
-- the symmetric algorithm used to encrypt the value
encSymmKey [2] BIT STRING OPTIONAL,
-- the (encrypted) symmetric key used to encrypt the value
keyAlg [3] AlgorithmIdentifier OPTIONAL,
-- algorithm used to encrypt the symmetric key
valueHint [4] OCTET STRING OPTIONAL,
-- a brief description or identifier of the encValue content
-- (may be meaningful only to the sending entity, and used only
-- if EncryptedValue might be re-examined by the sending entity
-- in the future)
encValue BIT STRING }
-- the encrypted value itself
KeyGenParameters ::= OCTET STRING
id-regCtrl-oldCertId OBJECT IDENTIFIER ::= { id-regCtrl 5 }
--with syntax:
OldCertId ::= CertId
CertId ::= SEQUENCE {
issuer GeneralName,
serialNumber INTEGER }
id-regCtrl-protocolEncrKey OBJECT IDENTIFIER ::= { id-regCtrl 6 }
--with syntax:
ProtocolEncrKey ::= SubjectPublicKeyInfo
-- Registration Info in CRMF
id-regInfo OBJECT IDENTIFIER ::= { id-pkip 2 }
id-regInfo-utf8Pairs OBJECT IDENTIFIER ::= { id-regInfo 1 }
--with syntax
UTF8Pairs ::= UTF8String
id-regInfo-certReq OBJECT IDENTIFIER ::= { id-regInfo 2 }
--with syntax
CertReq ::= CertRequest
END
```