目　次

前　言

引　言

1 范围

本规范适用于电子政务/电子商务基于密码技术的数字证书认证系统的设计、建设、检测、运营及管理，规范数字证书认证系统中密码协议的标准化应用，推动数字证书认证系统密码协议的互连互通和相互认证。对于组织或机构内部使用的数字证书认证系统密码协议的建设、运营及管理，可参考使用。

同时本规范还可为安全产品生产商提供产品和技术的准确定位和标准化的参考，提高安全产品的可信性和互操作性。

2 规范性引用文件

下列文件中的条款通过本规范的引用而成为本规范的条款。凡是注明日期的引用文件，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规范，然而，鼓励根据本规范达成协议的各方研究是否可使用这些文件的最新版本。凡是不注明日期的引用文件，其最新版本适用于本规范。

下列标准所包含的条文，通过在本规范中引用而构成为本规范的条文。考虑到标准的修订，使用本规范时应研究使用下列标准最新版本的可能性。

国家密码管理局 《证书认证系统密码及其相关安全技术规范》

GB/T 20518－2006 《公钥基础设施 数字证书格式》

GB/T 19713—2005 《公钥基础设施 在线证书状态协议》

GB/T 19714—2005 《公钥基础设施 证书管理协议》

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（idt ITU-T X.509 2000 ｜ ISO/IEC 9594-8）

GB/T16262.1—2002 信息技术 抽象语法记法1（ASN.1）:基本记法规范 （IDT ISO/IEC 8824-1：2002）

公钥密码基础设施应用技术体系 标识规范

GM/T AAAAA SM2密码使用规范

GM/T BBBBB SM2加密签名消息语法规范

GM/T CCCCC 标识规范

RFC3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List(CRL) Profile

3 术语和定义

3.1 认证机构证书 authority certificate

签发给证书认证机构的证书。

3.2 证书认证机构 certificate authority（CA）

又称为认证中心或CA中心，它是被用户所信任的签发公钥证书及证书撤销列表的管理机构。

3.3 CA撤销列表 certificate authority revocation list（CARL）

标记已经被撤销的CA的公钥证书的列表，表示这些证书已经无效。

3.4 证书认证系统 certificate authentication system

对生存周期内的数字证书进行全过程管理的安全系统。

3.5 CA证书 CA certificate

由一个CA给另一个CA签发的数字证书。一个CA也可以为自己签发证书，这是一种自签名的证书。

3.6 证书认证路径 certification path

在目录信息树中对象证书的一个有序的序列。路径的初始节点是最初待验证对象的公钥，可以通过路径获得最终的顶点的公钥。

3.7 证书策略 certificate policy

是一个指定的规则集合，它指出证书对于具有普通安全需求的一个特定团体和（或）具体应用类的适

用性。例如，一个特定的证书策略可以指出一个类型的证书对在一定的价格幅度下商品交易的电子数据处理的认证的适用性。

3.8　证书撤销列表 certificate revocation list（CRL）

标记一系列不再被证书发布者认为有效的证书的签名列表。

3.9　证书验证 certificate validation

确定证书在指定的时间内是否有效的过程。证书验证包括有效期验证、签名验证以及证书状态的检验。

3.10　证书撤销列表分布点 certificate revocation list distribution point

一个目录条目或其它的证书撤销列表分布源，一个通过证书撤销列表分布点发布的证书撤销列表，可以包括由一个 CA 发布的所有证书中的一个证书子集的撤销条目，也可以包括全部证书的撤销条目。

3.11　证书序列号 certificate serial number

在一个认证机构所签发的证书中用于唯一标识数字证书的整数。

3.12　数字证书 digital certificate

由认证权威数字签名的包含公开密钥拥有者信息、公开密钥、签发者信息、有效期以及一些扩展信息的数字文件。

3.13　公钥基础设施 public key infrastructure (PKI)

能够对公私钥对进行管理，支持身份验证、保密性、完整性以及不可否认性服务的信息安全基础设施。

3.14　证书注册机构 registration authority（RA）

证书认证体系中的一个组成部分，它是接收用户证书及证书撤销列表申请信息、审核用户真实身份、为用户颁发证书的管理机构。

3.15　相关协议

本规范涉及的相关协议指数字证书认证系统中涉及到密码技术的安全协议，特别是证书认证系统使用的有关安全协议。这些协议包括用户终端同 RA 之间的安全协议；RA 同 CA 之间的安全协议；CA 同 KM 之间的安全协议；CA 同 LDAP 服务之间的安全协议；CA 同 OCSP 服务之间的安全协议；用户同 LDAP 服务之间的安全协议；用户同 OCSP 服务之间的安全协议等。

3.16　协议报文

本规范给出的协议报文定义为安全协议的报文格式内容，对“相关协议”中给出的安全协议给出的具体表述。

3.17　密码服务接口

本规范给出的密码服务接口定义为协议中实际应用的密码接口，包括对称密钥密码接口、非对称密钥密码接口、数据摘要算法接口、数字签名接口、数字信封接口、证书应用接口以及其他有关应用接口。

3.18　密码报文语法

本规范给出的密码报文语法是指数字证书认证系统密码协议的具体应用语法。包括加密数据报文语法、摘要数据报文语法、数字签名报文语法和数字信封报文语法等。

4　缩略语

下列缩略语适用于本标准：

DIT	目录信息树系统(Directory Information Tree)
KM	密钥管理系统(Key Management)
OCSP	在线证书状态查询协议(Online Certificate Status Protocol)
OID	对象标识符(Object Identifier)
SOCSP	简明在线证书状态查询协议(Simple Online Certificate Status Protocol)
TBS	待签名的（证书）(To Be Signed)

5　相关协议

5.1 概述及协议流程

5.1.1 内容概述

本规范以安全协议为主体，给出了数字证书认证系统的相关密码协议。凡涉及互联互通的协议应作为规范性协议，例如 CA 与 KM 之间的协议、RA 同用户载体之间的协议以及证书签发、证书验证、证书查询协议等需要作统一规范，以促进我国数字证书认证系统密码协议的标准化体系建设。对不涉及互联互通的内部操作协议，本规范给出了基本要求，提供了技术支持，力求统一，但可不作强制。

本规范给出了与协议直接相关的格式、语法等内容。其中凡涉及 RSA 密码算法的，其密钥结构遵循 PKCS#1 规范，其封装结构遵循 PKCS#7 规范；凡涉及 SM2 算法的，其密钥结构遵照 GM/T AAAAA SM2 密码使用规范，其封装结构遵循 GM/T BBBBB SM2 加密签名消息语法规范。凡涉及对象标识符的，遵循 GM/T CCCCC 标识规范。

5.1.2 协议流程

本规范定义的相关协议流程用以下图 1 给出：

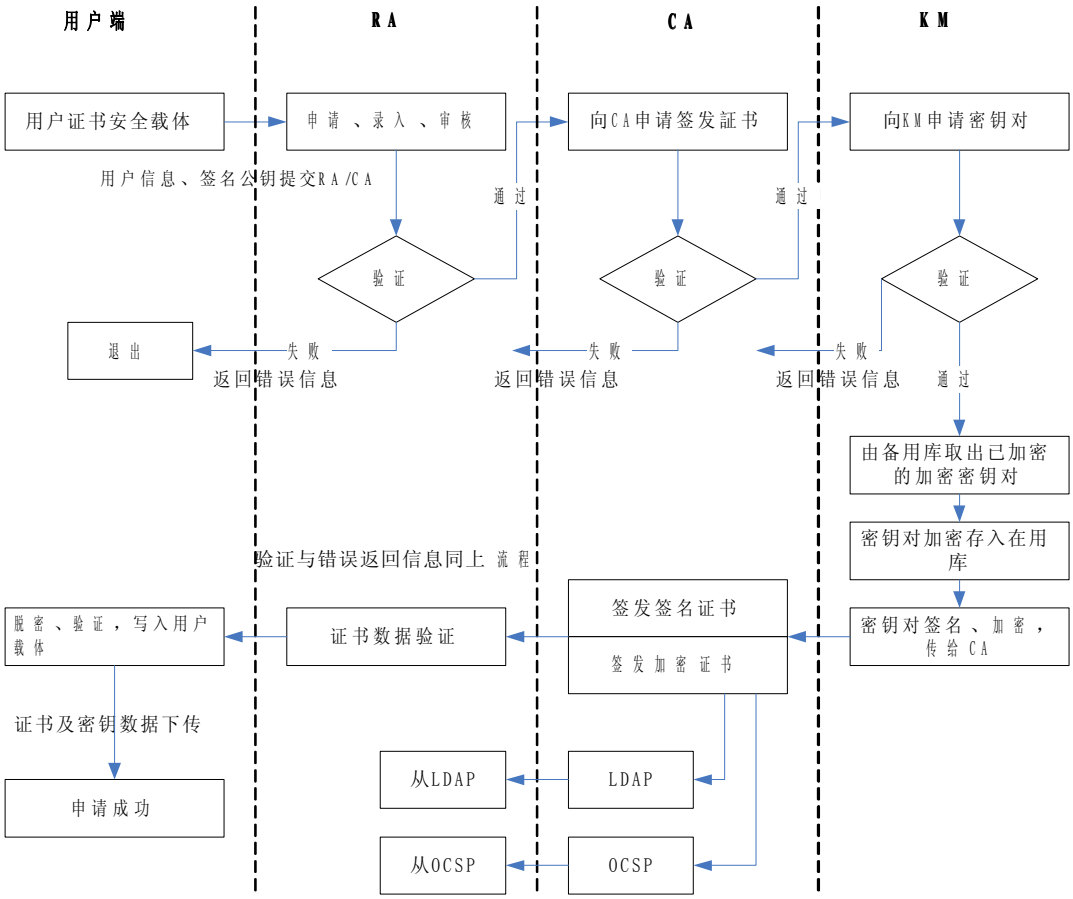

图 1 协议流程图

非实时发放证书的协议流程参见附录 D。

5.1.2.1　RA 同客户端间协议流程

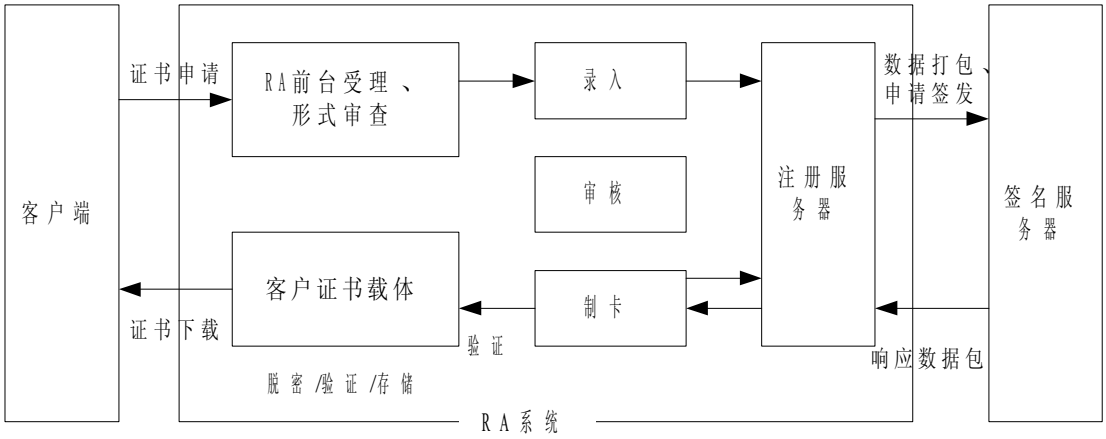

图 2 RA 与客户端协议流程

5.1.2.2　RA 同 CA 间协议流程

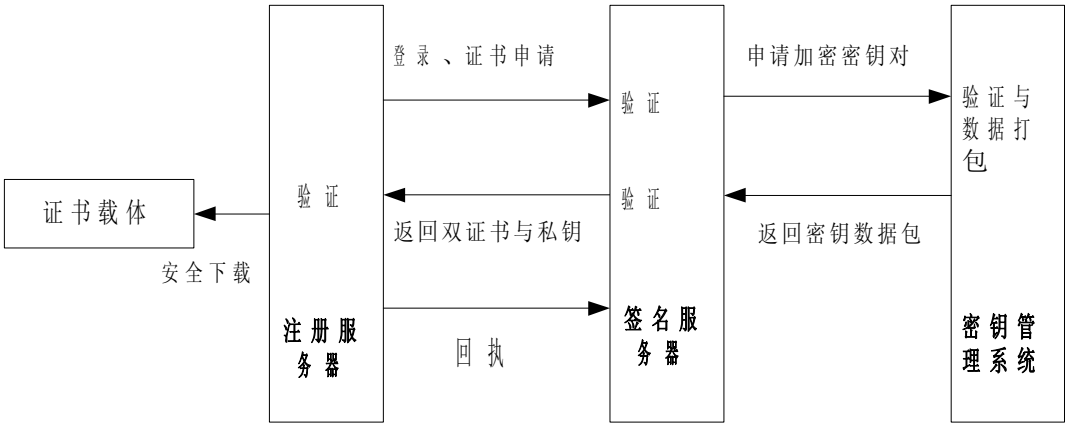

图 3 RA 与 CA 协议流程

5.1.2.3　CA 同 KM 间协议流程

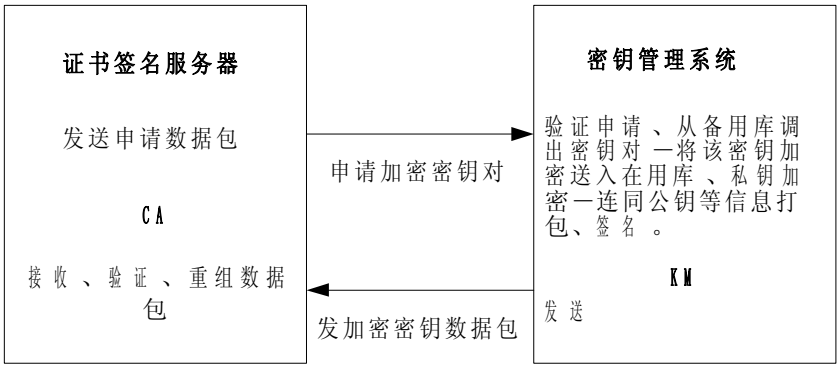

图 4 CA 与 KM 协议流程

5.1.2.4　CA 同与发布系统间协议流程

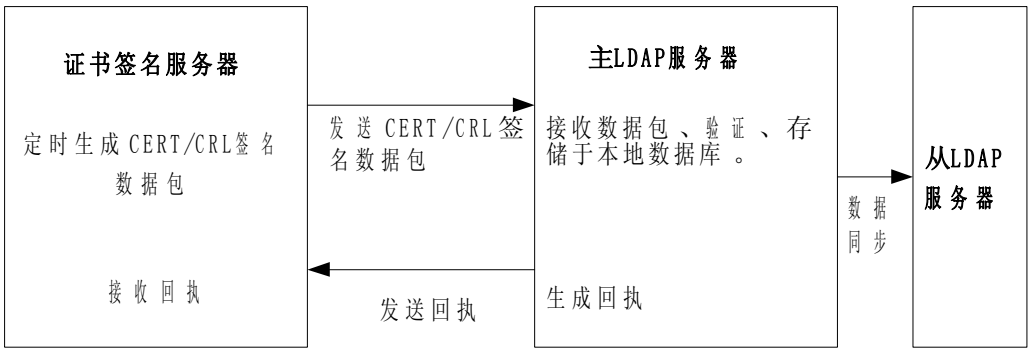

图 5 CA 与发布系统协议流程

5.1.2.5　用户与目录服务间协议流程

参见 LDAP 相关协议标准。

5.1.2.6　CA 与证书状态查询服务间协议流程

参见 GB/T 19713-2005《信息技术 安全技术 公钥基础设施 在线证书状态协议标准》。

参见简化在线证书状态协议 SOCSP 标准。

5.1.2.7　用户与证书状态查询服务间协议流程

参见 GB/T 19713-2005《信息技术 安全技术 公钥基础设施 在线证书状态协议标准》。

参见简化在线证书状态协议 SOCSP 标准。

SOCSP 一般用于运行比较频繁的应用服务器对访问者身份证书有效性的验证，采用该协议可以提高验证的效率。

5.2　CA 与 KM 系统间相关协议

5.2.1　概述

KM 系统接收 CA 系统的密钥服务请求，将处理结果返回给 CA。本规范中的密钥服务协议包括申请密钥对、恢复密钥对和撤销密钥对三个，每个服务都按照请求－响应的步骤执行。

请求：请求由 CA 提出，发送到 KM。CA 在生成用户加密证书、更新加密证书或者撤销加密证书时，首先组织密钥服务请求，发送到 KM，并延缓自身的事务处理过程，等待 KM 响应返回。

响应：响应由 KM 发起，发送到 CA。KM 在接收到来自 CA 的请求后，检查确定请求合法性，处理服务请求，并将结果返回给 CA。

整个服务过程可以使用图 6 示意表示。

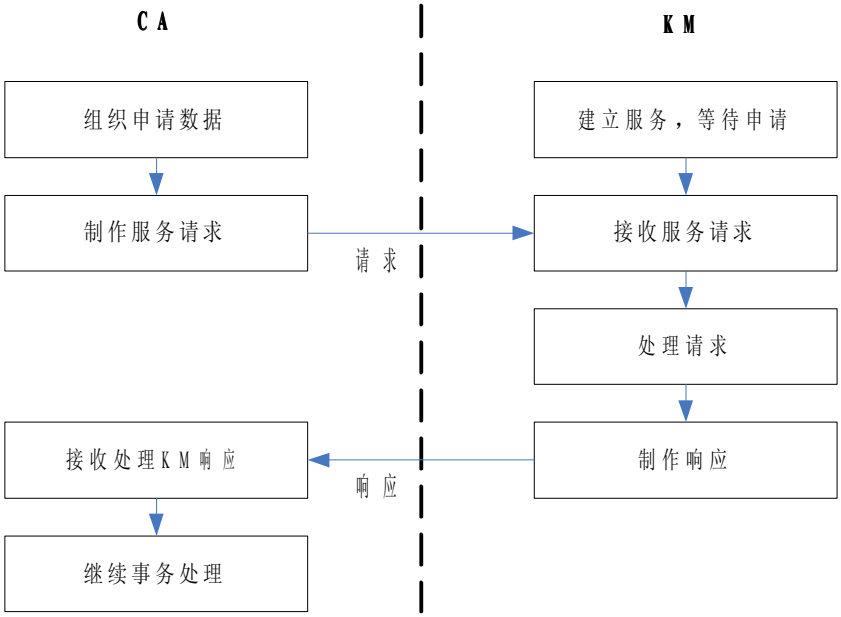

图 6 CA 和 KM 通讯过程

下面的协议内容将按照这个框架进行。

5.2.2 协议内容

a) 请求

也称密钥服务请求，包含 CA 请求的类型、性质以及特性数据等，该请求将被发送到 KM 并得到服务。服务请求包括如下数据：

——协议版本(当前版本为 2)；

——服务请求标识符；

——CA 标识符；

——扩展的请求信息；

——请求信息的签名。

b) 响应

指 KM 对来自 CA 请求的处理响应。KM 的响应包括如下数据：

——协议版本(当前版本为 2)；

——响应标识符；

——KM 标识符；

——响应信息；

——响应信息的签名。

c) 异常情况

当 CA 系统和 KM 系统任何一方处理发生错误时，均需要向对方发送错误信息。错误可以是下列几类：

——验证请求失败：KM 验证来自 CA 证书或 CA 请求数据失败，CA 收到后应重新进行申请。

——内部处理失败：KM 处理 CA 请求过程中发生内部错误，通知 CA 该请求处理失败，需要重新申请；

本规范采用抽象语法表示法（ASN.1）来描述具体协议内容。如果无特殊说明，默认使用 ASN.1 显式标记。

引用的其他术语还有：Extensions，CertificateSerialNumber，SubjectPublicKeyInfo，Name，AlgorithmIdentifier，CRLReason。其定义参见 RFC3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List(CRL) Profile。其中 SubjectPublicKeyInfo 在使用 SM2

算法的时候，其定义参见 GM/T BBBBB 《SM2 加密签名语法规范》。

5.2.3 密钥申请协议

5.2.3.1 请求数据格式

CA 请求的基本格式如下：

```
CARequest ::= SEQUENCE {
    ksRequest           KSRequest,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      OCTET STRING
}
```

其中：

```
KSRequest ::= SEQUENCE {
    version         Version DEFAULT v2,
    caName          EntName,
    requestList     SEQUENCE OF Request,
    requestTime     GeneralizedTime,
    taskNo          INTEGER
}
Version ::= INTEGER { v2(1)}
EntName::= SEQUENCE {
    hashAlgorithm   AlgorithmIdentifier,
    entName         GeneralName,
    entPubKeyHash   OCTET STRING,
    serialNumber    CertificateSerialNumber
}
CertificateSerialNumber::=  INTEGER
Request ::=CHOICE{
    applyKeyReq     [0] IMPLICT ApplyKeyReq,
    restoreKeyReq   [1] IMPLICT RestoreKeyReq,
    revokeKeyReq    [2] IMPLICT RevokeKeyReq
}
```

;(注：前面已直接定义为 GerneralTime，此处应该不再表述)

5.2.3.2 KSRequest 及其结构解释

KSRequest 包含了请求语法中的重要信息，本节将对该结构作详细的描述和解释。

a) 版本

本项描述了请求语法的版本号，当前版本为 2，取整型值 1。

b) 请求者标识符

本项描述了请求者标识符。entName 是申请者的唯一名称，该值由实际运行 CA 和 KM 约定。

entName 结构中，entPubKeyHash 是申请者公钥的哈希值。该值将通过对发布者证书中的主体公钥字段(不含标记和长度)进行计算。hashAlgorithm 字段用来指明这些哈希计算所使用的哈希算法。serialNumber 是申请者的证书序列号。

c) 请求类型

本项描述了请求包类型，当值为 applyKey 时，表明该包为申请密钥申请包，为 restoreKey 时表明该

包为恢复密钥申请包，值为 revokeKey 时表明该包为撤销密钥申请包。

本项的取值决定了下面一项——详细请求子包 Request 的取值。

详细请求子包的格式如下：

```
Request ::= CHOICE {
    applykeyreq     [0] IMPLICT ApplyKeyReq,
    restorekeyreq   [1] IMPLICT RestoreKeyReq,
    revokekeyreq    [2] IMPLICT RevokeKeyReq
}
```

上面三种数据格式解释如下：

——ApplyKeyReq 包

ApplyKeyReq 包为密钥申请格式包，其具体格式如下：

```
ApplyKeyReq :: SEQUENCE {
    appKeyType      AlgType,
    appKeyLen       AppKeyLen,
    retAsymAlg      AlgType,
    retSymAlg       AlgType,
    retHashAlg      AlgType,
    appUserInfo     AppUserInfo
}
```

AlgType ::= AlgorithmIdentifier, 表明使用的非对称算法、对称算法、摘要算法等算法类型。其中，appKeyType 为要申请的加密密钥对的类型，retAsymAlg、retSymAlg、retHashAlg 分别为 KM 响应数据包中非对称算法、对称算法、摘要算法类型；

AppKeyLen ::= INTEGER，表示申请的密钥强度，如十进制 1024 表示申请 1024 位（bit）长度的密钥；

```
AppUserInfo::= SEQUENCE {
    userCertNo      CertificateSerialNumber,
    userPubKey      SubjectPublicKeyInfo,
    notBefore       GeneralizedTime,
    notAfter        GeneralizedTime,
    userName        [0] OCTET STRING OPTIONAL,
    dsCode          [1] FreeText OPTIONAL,
    extendInfo      [2] FreeText OPTIONAL
}
```

AppUserInfo 结构表示申请包中对应用户信息，依次为终端用户加密证书序列号、终端用户保护公钥、密钥有效起始时间、密钥截止时间、用户姓名、地区代码以及扩展信息。

对于同一个 CA，其申请时提交的用户加密证书序列号必须是唯一的，如果需要使用原有证书序列号再次申请密钥对，则应当先请求将原申请密钥对撤销。

userPubKey 为终端用户保护公钥，该公钥必须由终端用户证书载体生成，在本系统中用来在密钥传递中保护用户加密私钥。该公钥推荐采用终端用户签名公钥。

用户信息结构中，userName 为可选项，该项用来标识用户名称，当 CA 将该项提交给 KM 时，KM 应当将其保存到库中。

dsCode 为可选项，CA 可以使用该项标识密钥的地区属性，当 dsCode 存在时，KM 必须将其保存到库中。

extendInfo 扩展信息为可选项，用来表示 CA 需要往 KM 发送关于该用户的个性信息。比如要求 KM 按照区域管理密钥，CA 可以利用该域填写该用户的区域信息，KM 则可以读出该域值后应保存到库中。该域最大应能处理 100 字节数据。

——RestoreKeyReq 包

RestoreKeyReq 包为密钥恢复格式包，其具体格式如下：

```
RestoreKeyReq :: SEQUENCE {
    retAsymAlg  AlgType,
    retSymAlg   AlgType,
    retHashAlg  AlgType,
    userCertNo  CertificateSerialNumber,
    userPubKey  SubjectPublicKeyInfo,
}
```

AlgType ::= AlgorithmIdentifier, 表明使用的非对称算法、对称算法、摘要算法等算法类型。其中，retAsymAlg、retSymAlg、retHashAlg 分别为 KM 响应数据包中非对称算法、对称算法、摘要算法类型；

userCertNo 指定用户证书序列号；

userPubKey 指定用户签名公钥。

——RevokeKeyReq 包

RevokeKeyReq 包为密钥撤销格式包，其具体格式如下：

```
RevokeKeyReq :: SEQUENCE {
userCertNo  CertificateSerialNumber,
}
```

UserCertNo 指用户证书序列号。

d) 请求时间

本项描述请求生成时间，该时间即为 CA 产生请求的时间，采用 GeneralizedTime 语法表示。

e) (注，该项已被删除)任务序列号

本项描述了请求的任务序列号，该任务序列号是申请者用来区分多次申请时候的一个标识符，以确保 KM 和 CA 能正确关联请求－响应等过程。

任务序列号是一个整型值，KM 应能处理不大于二十字节的任务序列号，而 CA 应确保不使用大于二十字节的任务序列号。

5.2.4 响应

响应数据格式

KM 响应的基本格式如下：

```
KMRespond ::= SEQUENCE {
    ksRespond           KSRespond,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      OCTET STRING
}
```

其中：

```
KSRespond::= SEQUENCE {
    version         Version DEFAULT v2,
    KMName          entName,
    respondList     SEQUENCE OF Respond,
    respondTime     GeneralizedTime, (修改这里是为了与请求一致)
    taskNO          INTEGER
}
```

Version、entName、TaskNo 数据格式在前文已经解释。

```
Respond ::=CHOICE{
    applykeyRespond      [0] IMPLICT RetKeyRespond,
    restorekeyRespond    [1] IMPLICT RetKeyRespond,
    revokekeyRespond     [2] IMPLICT RevokeKeyRespond,
    errorPkgRespond      [3] IMPLICT ErrorPkgRespond
}
```

5.2.4.1 KSRespond 及其结构解释

KSRespond 包含了响应语法中的重要信息，本节将对该结构作详细的描述和解释。

a) 版本

本项描述了响应语法的版本号，当前版本为 2，取整型值 1。

b) 响应者标识符

本项描述了响应者标识符，该结构在前文已经给出，在本响应数据中，其成员分别取值响应者的名称、响应者公钥的哈希值、哈希算法以及响应者的证书序列号。

c) 响应类型

本项描述了响应包类型，当值为 applyRespond 时，表明该包为申请密钥响应包，为 restoreRespond 时表明该包为恢复密钥响应包，值为 revokeRespond 时表明该包为撤销密钥响应包。

本项的取值决定了下面一项——详细响应子包 Respond 的取值。

a) 详细响应子包

本项描述响应者请求中的详细响应子包，每个子包的格式如下：

```
Respond ::= CHOICE {
    applyKeyRespond      [0] IMPLICT retKeyRespond,      restoreKeyRespond   [1]  IMPLICT
retKeyRespond,
    revokeKeyRespond     [2] IMPLICT RevokeKeyRespond,
    errorPkgRespond      [3] IMPLICT ErrorPkgRespond
}
```

上面共有三种数据格式，分别解释如下：

——retKeyRespond 包

retKeyRespond 包为密钥响应格式包，在处理申请密钥、恢复密钥申请时响应申请者的，其具体格式如下：

```
retKeyRespond ::= SEQUENCE {
    userCertNo  CertificateSerialNumber,
    retPubKey   SubjectPublicKeyInfo,
    retPriKey   dataEnvelope
}
dataEnvelope ::= SignedAndEnvelopedData (data)
```

UserCertNo 指用户加密证书序列号，该项从 CA 申请包中取值；

retPubKey 是返回给申请者的用户加密公钥数据；

retPriKey 是返回给申请者的用户加密私钥数据信封，即对私钥作带签名的加密信封。

——RevokeKeyRespond 包

RevokeKeyRespond 包为密钥撤销响应格式包，在处理密钥撤销时响应申请者的其具体格式如下：

```
RevokeKeyRespond ::= SEQUENCE {
    userCertNo  CertificateSerialNumber,
}
```

UserCertNo 指定用户加密证书序列号，该项值取自申请包。

——ErrorPkgRespond 包

ErrorPkgRespond 包为错误包，在处理密钥服务请求出错时，KM 使用本包响应申请者其具体格式如下：

```
ErrorPkgRespond ::= SEQUENCE {
    (注，该项前面已没有)errNo            INTEGER,
    errDesc          [0] FreeText OPTIONAL
}
```

UserCertNo 指定用户加密证书序列号，该项值取自申请包。

b) 响应时间

本项描述响应生成时间，该时间即为 KM 产生响应的时间，采用 GeneralizedTime 语法表示。

c) 任务序列号

本项描述了响应的任务序列号，该任务序列号值取自申请者数据包。

5.3 CA 与 LDAP 服务间相关协议

5.3.1 协议概述

证书与证书撤销链发布是指 CA 把新签发的证书与证书撤销链送到 LDAP 目录服务器，以供用户查询、下载。

现有的 LDAP 服务产品，都会提供服务管理与客户端接口。设计好数据存储模板后，只要调用客户端接口，就可以把数据传入 LDAP 服务产品，数据的传输采用 LDAP 协议。但是，按照国家密码管理局《证书认证系统密码及其相关安全技术规范》规定：“证书/证书撤销列表存储发布系统采用主、从结构的目录服务器，签发完成的数据直接写入主目录服务器，然后由主目录服务器的主从映射功能自动映射到从目录服务器中。主、从目录服务器通常配置在不同等级的安全区域。用户只能访问从目录服务器”。由此可见，在 CA 到 LDAP 服务间的数据传输安全上有明确要求。

借鉴 PKIMessage 流程设计经验，CA 到 LDAP 服务的发布协议设计如下：

5.3.2 发布协议

传输说明：

```
"Content-Type : application/pkixissue\r\n"
"Content-Length : 1222\r\n"
"\r\n"
```

消息包结构说明：

```
PkixIssue ::= SEQUENCE {
PkixIssueInfo TBSISSUE,发布内容
SignatureAlgorithm SignatureAlgorithmIdentifier,
CASignature Signature
}
TBSISSUE SEQUENCE::={
version     Version,-版本号，目前为 1
type        INTEGER,
                -0：向 LDAP 或 OCSP 更新根证书
                -1：向 LDAP 发送证书，
                -2：向 LDAP 发送作废证书序列号，
                -3：向 LDAP 发送作废证书链，
                -4：向 OCSP 发送证书状态
                -8：向 LDAP 发送交叉认证证书，LDAP 自己用根证书与此包中的第一个证书组
```

成 P7 证书发布链

transsNonce OCTET STRING OPTIONAL,包内随机数，来回要一致

nummber INTEGER OPTIONAL ,包内证书或证书状态或证书撤销链数目

time GeneralizedTime,接收方比较此时间，根据约定时间延迟确定是否接收包内内容

cert [0] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL，如果是证书，放在这项中，如果是更新根证书将发布 4 个证书，依次为 OldWithNew、 NewWithOld 、NewWithNew 签名证书、NewWithNew 加密证书

certstatue [1] SEQUENCE OF SEQUENCE {

certId CertID,

BeforeTime GeneralizedTime,

EndTime GeneralizedTime,

statue INTEGER,0 有效，1 无效，无效时有原因项

statueTime GeneralizedTime,

statueReasonRode CRLReason OPTIONAL,

}

mCRL [2] SEQUENCE OF SEQUENCE {

stateOrProvinceNum INTEGER, 区域区别号，CRL 发布点可以很多，发布内容可以不同，只有唯一地点的为 0

CLRSegment INTEGER, 证书链的块号，为防证书链太大，一般按证书号分块，同块内证书在同一条链中

CRLNumber INTEGER, 证书链基本序列号

DeltareCRLNumber INTEGER OPTIONAL, 证书链增量序列号

CertificateList SEQUENCE { 作废证书链

tbsCertList TBSCertList,

signatureAlgorithm AlgorithmIdentifier,

signatureValue BIT STRING

}

}

}

signatureAlgorithm AlgorithmIdentifier OPTIONAL,签名算法

signatureValue BIT STRING OPTIONAL，CA 的签名，注意：若是根证书更新包，这里的签名是新根的签名，接收方需一级级验证，以确保新根合法

}

收到消息的回应包：

```
PkixIssueResponse ::= SEQUENCE {
  PkixIssueResponseInfo TBSISSUEResponse,
  SignatureAlgorithm SignatureAlgorithmIdentifier,
  ResponseSignature Signature
}
TBSISSUEResponse SEQUENCE::={
  version       Version,版本号，目前为 1
  type      INTEGER,
                    0：向 LDAP 或 OCSP 更新根证书
                    1：向 LDAP 发送证书，
```

2：向 LDAP 发送作废证书序列号，

3：向 LDAP 发送作废证书链，

4：向 OCSP 发送证书状态

8：向 LDAP 发送交叉认证证书，LDAP 自己用根证书与此包中的第一个证书组成 P7 证书发布链

transNonce OCTET STRING OPTIONAL,包内随机数，来回要一致

nummber INTEGER OPTIONAL ,包内证书或证书状态或证书撤销链数目

time GeneralizedTime 接收方时间

ResponseStatue INTEGER 0 为正常接收，1 为未接收，请下次重发

}

说明：如果 CA 收到回应后验证签名不过或传输随机数不同，此次发布失败，下次要重新打包再发布。

5.4 用户与 LDAP 服务间相关协议

5.4.1 协议概述

LDAP V2 作为标准已基本停用，本节主要讨论 V3 协议；

LDAP V3 协议也不是一个协议，而是一个协议族；

2251——LDAP V3 核心协议，定义了 LDAP V3 协议的基本模型和基本操作；

RFC 2252——定义了 LDAPV3 中的基本数据模式（Schema）（包括语法、匹配规则、属性类型和对象类）以及标准的系统数据模式；

RFC 2253——定义了 LDAP V3 中的分辨名（DN）表达方式；

RFC 2254——定义了 LDAP V3 中的过滤器的表达方式；

RFC 2255——LDAP 统一资源地址的格式；

RFC 2256——在 LDAPV3 中使用 X.500 的 Schema 列表；

RFC 2829——定义了 LDAPV3 中的认证方式；

RFC 2830——定义了如何通过扩展使用 TLS 服务；

RFC 1823——定义了 C 的 LDAP 客户端 API 开发接口；

RFC 2829 --- Authentication Methods for LDAP;

RFC 2696 LDAP Control Extension for Simple Paged Results Manipulation;

RFC 2222 -- -Simple Authentication and Security Layer (SASL)″;

RFC 1487 X.500 Lightweight Directory Access Protocol (covers version 1, now obsolete);

RFC 1777 X.500 Lightweight Directory Access Protocol (covers LDAPv2);

RFC 1798 Connection-less Lightweight X.500 Directory Access Protocol;

RFC 1823 The LDAP Application Program Interface;

RFC 2247 Using Domains in LDAP/X.500 Distinguished Names;

RFC 2377 Naming Plan for Internet Directory-Enabled Applications;

LDAP 目录可以用来存储多种信息，作为 PKI 的核心组件其作用主要用来存放证书与证书撤销列表，供用户查询。

RFC2251 核心协议中定义了 LDAP 消息协议数据单元 LDAPMessage 如下：

```
LDAPMessage ::=SEQUENCE {
  messageID       MessageID,
  protocolOp      CHOICE {
bindRequest BindRequest,
bindResponse BindResponse,
unbindRequest UnbindRequest,
searchRequest SearchRequest,
searchResEntry SearchResultEntry,
```

searchResDone SearchResultDone,
searchResRef SearchResultReference,
modifyRequest ModifyRequest,
modifyResponse ModifyResponse,
addRequest AddRequest,
addResponse AddResponse,
delRequest DelRequest,
delResponse DelResponse,
modDNRequest ModifyDNRequest,
modDNResponse ModifyDNResponse,
compareRequest CompareRequest,
compareResponse CompareResponse,
abandonRequest AbandonRequest,
extendedReq ExtendedRequest,
extendedResp ExtendedResponse,
controls [0] Controls OPTIONAL
},
}
MessageID ::= INTEGER (0 .. maxInt)
maxInt INTEGER ::= 2147483647 -- (2^^31 - 1) --
LDAP 消息(LDAPMessage)的功能是给所有的协议交换定义一个包含通用字段的封装。
除了在上面定义的 LDAPMessage，在定义协议操作时，也使用下面这些定义。
LDAPString ::= OCTET STRING
LDAPString 是一种符号上的方便表示，它表明尽管 LDAPString 是一种用字符串类型来编码的串，但实际上该串能使用的合法字符集由 IA5 字符集限定。
LDAPDN ::= LDAPString
RelativeLDAPDN ::= LDAPString
一个 LDAPDN 和一个 RelativeLDAPDN 被独立地定义，分别代表一个标识名和一个相对标识名：
AttributeValueAssertion ::=
SEQUENCE {
attributeType AttributeType,
attributeValue AttributeValue
}
AttributeValueAssertion 的类型定义与 X.500 目录标准类似
AttributeType ::= LDAPString
AttributeValue ::= OCTET STRING
注意这里的 AttributeType，如果服务器端拥有某个属性的文本名，则在查询结果中返回文本名，在这里文本名字（“OrganizationName”）优先于点数方式（“2.5.4.10”）。当然服务器端应拥有一个较全的属性名列表，且便于客户端知道。附录的例子中可以清楚地看到。
一个类型为 AttributeValue 的域的值用目录中的 AttributeValue 类型的一个八位编码串来表示。
LDAPResult ::=SEQUENCE {
resultCode ENUMERATED {
success (0),
operationsError (1),

```
protocolError (2),
timeLimitExceeded (3),
sizeLimitExceeded (4),
compareFalse (5),
compareTrue (6),
authMethodNotSupported (7),
strongAuthRequired (8),
-- 9 reserved --
referral (10), -- new
adminLimitExceeded (11),  -- new
unavailableCriticalExtension (12),  -- new
confidentialityRequired (13),  -- new
saslBindInProgress (14),  -- new
noSuchAttribute (16),
undefinedAttributeType (17),
inappropriateMatching (18),
constraintViolation (19),
attributeOrValueExists (20),
invalidAttributeSyntax (21),
-- 22-31 unused --
noSuchObject (32),
aliasProblem (33),
invalidDNSyntax (34),
-- 35 reserved for undefined isLeaf --
aliasDereferencingProblem (36),
-- 37-47 unused --
inappropriateAuthentication (48),
nvalidCredentials (49),
nsufficientAccessRights (50),
busy (51),
unavailable (52),
unwillingToPerform (53),
loopDetect (54),
-- 55-63 unused --
namingViolation (64),
bjectClassViolation (65),
notAllowedOnNonLeaf (66),
notAllowedOnRDN (67),
entryAlreadyExists (68),
objectClassModsProhibited (69),
-- 70 reserved for CLDAP --
affectsMultipleDSAs (71),  -- new
-- 72-79 unused --
other (80)
```

```
},
-- 81-90 reserved for APIs --
matchedDN       LDAPDN,
errorMessage        LDAPString,
referral        [3] Referral OPTIONAL
}
```

LDAPResult 是本协议中从服务器返回给客户用以指明操作成功或失败的结构。对不同的请求，服务器应该返回包含 LDAPResult 结构中的不同域的回应给客户，来指明协议操作请求的最终状态。对服务器来说，这个结构中的 errorMessage 域应该用来返回一个包含可读文本的错误诊断的 ASCII 串给客户。由于这个错误诊断不是标准的，在实现中不应该依赖这些返回值。如果服务器不返回一个文本的错误诊断，LDAPResult 结构中的 errorMessage 应该包含一个长度为零的字符串。

对于 noSuchObject，aliasProblem，invalidDNSyntax，及 aliasDereferencingProblem 这些 resultCodes，相应的 matchedDN 域应该设为在 DIT 中最为匹配的条目，而且应该是所提供名字的简短格式。或者，如果别名已被丢弃，应该返回结果名字。在其他情况下，matchedDN 域都应该设为 NULL DN(也就是长度为零的字符串)。

客户和服务器之间的协议会晤，包括绑定、解除绑定、查询、插入、修改、删除、丢弃等。依据国家密码管理局《证书认证系统密码及其相关安全技术规范》规定，用户只能访问从 LDAP，所以 LDAP 必须拒绝用户的插入、修改、删除功能。协议会晤中通用部分介绍如下：

绑定操作的功能是在客户和服务器之间初始化一个协议会晤，并允许服务器对客户进行认证。绑定请求定义如下：

```
BindRequest ::=[APPLICATION 0] SEQUENCE {
                            version    INTEGER (1 .. 127),
                            name       LDAPDN,
                            authentication  uthenticationChoice
}
AuthenticationChoice ::= CHOICE {
simple [0] OCTET STRING,
 -- 1 and 2 reserved
sasl [3] SaslCredentials
}
SaslCredentials ::= SEQUENCE {
mechanism LDAPString,
credentials OCTET STRING OPTIONAL
}
```

绑定请求的参数是：

——版本：一个版本号指定本协议会晤中所使用协议的版本。本文档描述的是 LDAP 版本 3。注意，由于没有版本选择谈判，客户只须把这个参数设为它所希望的值。如果客户端描述的是 LDAP 版本 2，服务器端不应返回版本 3 所加内容。如果客户端没有绑定请求（V3 版可以没有绑定），服务器端必须认定客户端支持 V3 版协议。

——名称：客户希望绑定的目录对象的名称。这个域可以是空值(一个长度为零的字符串)，用来表示匿名绑定。

——认证：与 LDAP 版本 2 相比，版本 3 在认证上有所加强。SASL（Simple Authentication and Security Layer)是一个附加在面向连接协议上的框架，它提供了更高一级的安全认证机制。但作为从 LDAP，客户查询的认证必要性不大。

绑定操作需要如下定义的绑定回应：

```
BindResponse ::= [APPLICATION 1] SEQUENCE {
COMPONENTS OF LDAPResult,
serverSaslCreds [7] OCTET STRING OPTIONAL
}
```

放弃绑定操作的功能是临时终止一个协议会晤，放弃绑定操作定义如下：

```
UnbindRequest ::= [APPLICATION 2] NULL
```

没有定义放弃绑定操作回应。在收到一个放弃绑定请求后，服务器可以假设发送请求的客户已终止该会晤，所有已收到的还未处理的请求可以被丢弃。

丢弃操作的功能是允许客户请求服务器终止一个已发出的操作请求。丢弃操作定义如下：

```
AbandonRequest ::= [APPLICATION 16] MessageID
```

在丢弃操作中没有定义相应的回应.在发送一个丢弃操作后,一个客户可能期望丢弃请求中包含的消息 ID 标识的操作已被丢弃。如果服务器在把一个查找操作的回应发送给客户时，收到对该查找操作的丢弃请求，服务器应停止发送回应，并立即终止该查找操作。

5.4.2 证书查询与下载协议

查找操作允许客户请求代表他的服务器进行一次查找。查找请求定义如下：

```
SearchRequest ::=[APPLICATION 3] SEQUENCE {
            baseObject    LDAPDN,
            scope         ENUMERATED {
                           baseObject            (0),
                           singleLevel           (1),
                           wholeSubtree          (2)
                          },
            derefAliases  ENUMERATED {
                            neverDerefAliases     (0),
                            derefInSearching      (1),
                            derefFindingBaseObj   (2),
                            derefAlways           (3)
                          },
            sizeLimit     INTEGER (0 .. maxInt),
            timeLimit     INTEGER (0 .. maxInt),
            attrsOnly     BOOLEAN,
            filter        Filter,
            attributes    SEQUENCE OF AttributeType
    }
Filter:: = CHOICE {
            and                [0] SET OF Filter,
            or                 [1] SET OF Filter,
            not                [2] Filter,
            equalityMatch      [3] AttributeValueAssertion,
            substrings         [4] SubstringFilter,
            greaterOrEqual     [5] AttributeValueAssertion,
            lessOrEqual        [6] AttributeValueAssertion,
```

```
            present             [7] AttributeType,
            approxMatch         [8] AttributeValueAssertion,
extensibleMatch     [9] MatchingRuleAssertion
        }
SubstringFilter:: =SEQUENCE {
            type            AttributeType,
                  -- at least one must be present
            substrings          SEQUENCE OF CHOICE {
                initial         [0] LDAPString,
                any             [1] LDAPString,
                final           [2] LDAPString
            }
}
MatchingRuleAssertion:: = SEQUENCE {
matchingRule [1] MatchingRuleId OPTIONAL,
type [2] AttributeDescription OPTIONAL,
matchValue [3] AssertionValue,
dnAttributes [4] BOOLEAN DEFAULT FALSE
}
```

查找请求的参数是：

——baseObject： 一个相对于要进行查找操作的基本对象条目。

——scope： 指示要查找的范围。该域可能的语义上的值与目录查找操作中的范围域的语义值是一致的。

——derefAliases：指示在操作中别名对象该如何处理。该域可能的语义上的值按照升序的顺序进行。

——neverDerefAliases：在查找或定位查找的基本对象时不丢弃别名引用

——derefInSearching： 在查找过程中，丢弃基本对象的下一级的别名引用，但对基本对象进行定位时，不丢弃别名引用。

——derefFindingBaseObject： 在定位基本对象时，丢弃别名引用，但在查找基本对象的下一级时，不丢弃别名引用。

——derefAlways： 在定位基本对象和查找基本对象的下一级时都丢弃别名引用。

——sizelimit： 一个 sizelimit 限制了可以返回的最大查找结果的条目数量。如果该域的值为 0，表示不限制查找的 sizelimit。.

——timelimit： 一个 timelimit 限制了一个查找最多允许的时间(以秒计算)。如果该域的值为 0，表示不限制查找的 timelimit。

——attrsOnly： 指示在返回的查找结果中是否要包含属性类型和属性值，或者仅包含属性类型。如果该域设为 TRUE，仅返回属性类型(没有值)，如果该域设为 FALSE，同时返回属性类型和属性值。

——filter： 一个过滤器定义了一个必须满足的条件，以使该查找能匹配给定的条目。

——attributes： 要返回的查找结果中的每一个条目的属性列表。一个空的列表表示查找结果应该包含中每一个条目的所有属性。

在收到一个查找请求后，服务器返回如下定义的查找回应结果：

```
SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
objectName LDAPDN,
attributes PartialAttributeList
}
```

```
PartialAttributeList ::= SEQUENCE OF SEQUENCE {
type AttributeDescription,
vals SET OF AttributeValue
}
SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL
SearchResultDone ::= [APPLICATION 5] LDAPResult
```

在收到一个查找请求后,一个服务器将对 DIT 进行必要的查找。服务器将把一系列的回应返回给客户。这些回应包括:

a) 零个或多个查找回应，每个回应包含在查找时所找到的一个条目。在每个回应的后面跟着回应的序列号。

b) 一个包含操作成功或对所发生错误的详细描述的单个查找回应。

每个返回的条目必须包括所有的属性,以及在查找请求的 'attributes' 域中指定的所要求的关联的值.

注意一个 X.500 的"列表"操作可以通过具有检查 objectClass 属性是否存在这样一个过滤器的一个 LDAP 查找操作来模拟。同样, 一个 X.500 的"读"操作可以通过具有同样过滤器的基本对象的 LDAP 查找操作来模拟.

5.4.3 CRL 查询与下载协议

对于 LDAP 来说，证书或证书撤销列表都是一条条数据，只是存放在不同的分支下。用户查询下载的是证书还是 CRL，LDAP 根据“baseObject”与属性值查询满足条件的记录给予返回。

baseObject 是查找的基本条件，具体查找一条 CRL 要根据目录树的设计区别对待。目前常见的 CRL 目录有二种设计：

a) 根据证书号分段,如201段CRLLDAPDN存放“serialnumber＝？？？、ou=crl201、dc=isc、dc=com”,202段CRLLDAPDN存放“serialnumber＝？？？、ou=crl202、dc=isc、dc=com”。其中CRLNumber放到serialnumber中，分段号组合到ou中，具体的区别名放到dc中。

b) 含有增量CRLCRLLDAPDN存放“serialnumber=？？？,ou=dcrl,dc=isc,dc=com”。因为增量CRL的序列号是CRL基本序列号的累加，如基本号为1，增量号为2……9,下一个基本号为10，接下来的增量号为11……19,这个递增的号码不会重复，因此把这个递增的号码放入serialnumber。

对第一种情况，假设要查找 CRLNumber 为 100，分段号为 10 的 CRL，baseObject 可以这样组织:“serialnumber=100,ou=crl10,dc=isc,dc=com”，发送这样的请求就可获得具体的 CRL 列表。

对第二种情况，假设要查找 serialnumber 为 100，baseObject 可以这样组织:“serialnumber=100,ou=dcrl,dc=isc,dc=com”，发送这样的请求就可获得具体的 CRL 列表。对于一个应用来说，如果这个列表是基本表，时间也满足要求，通过这个表就可以知道证书是否作废；如果这个列表是增量表，时间也满足要求，通过这个表还不能知道证书是否作废，这时还要下载对应此表的基本表以及到目前表为止的增量表，经过每个表的检查方知具体证书的作废情况。

5.5 CA 与 OCSP/SOCSP 服务间相关协议

5.5.1 证书状态发布协议

CertID 结构用于标识特定的证书。

```
CertID ::= SEQUENCE {
    hashAlgorithm       AlgorithmIdentifier,
    issuerNameHash      OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash       OCTET STRING, -- Hash of Issuers public key
    serialNumber        CertificateSerialNumber }
```

关于证书状态的发布这里介绍二种方式，开发者可以根据熟悉情况自选一种。一种是 PKIXIssue 方式,

参见 CA 到 LDAP 服务的发布协议，其中含有发布到 OCSP 服务的解释。另一种是 PKIMessage 方式，下面是定义的格式。

当 CA 已经或将要作废一个特定的证书时，可以发布一个有关该事件(可能是将要发生的事件)的告示。

```
RevAnnContent ::= SEQUENCE {
status              PKIStatus,
certId              CertID,
willBeRevokedAt     GeneralizedTime,
badSinceDate            GeneralizedTime,
crlDetails          Extensions  OPTIONAL
-- 额外的 CRL 细节(如 crl number, reason, location 等)
}
```

CA 可以使用这样的告示来警告(或通知)一个申请者其证书将要(或已经)作废。这一消息通常用于作废请求并不是由相关证书的 subject 发起的情况下。

willBeRevokedAt 字段包含新的条目将增加到相关 CRLs 的时间。

5.5.2 SOCSP 证书状态查询协议

SOCSP 协议是参照国内拟制定的服务器对服务器间的简易 OCSP 有关标准。协议格式同 OCSP 协议，只是签名不再需要，实现时应考虑使用 HMAC 代替签名。

5.6 用户与 OCSP/SOCSP 服务间相关协议

5.6.1 OCSP 证书状态查询协议

在线证书状态协议（OCSP）使得应用程序可以获得所需要检验证书的状态。

这个协议描述了在应用程序检查证书状态和服务器提供状态之间所需要交换的数据。

一个 OCSP 请求包含以下数据：协议版本、服务请求、目标证书标识、可能被 OCSP 服务器处理的可选扩展。作为 OCSP 服务器要检测客户请求与服务策略之间的合理化。如果请求格式不对或服务器要求的客户信息不够，那么 OCSP 服务器将产生一个错误信息；否则的话，返回一个确定的回复。

OCSP 回复可以有几种类型。一个 OCSP 回复由回复类型和实际回复字节组成。有一种 OCSP 基本回复类型必须被所有的 OCSP 服务器和客户端支持。本章的其余部分都仅适用于这个回复类型。

所有确定的回复都应被数字签名。被用来签名回复信息的密钥必须是下列中的一个

——颁发所涉及证书的 CA

——一个被信任的 OCSP 服务器，它的公钥被请求者信任

——一个 CA 指派的 OCSP 服务器（被授权的 OCSP 服务器），它具有一张由 CA 直接颁发的用来表示此 OCSP 服务器可以为本 CA 发布 OCSP 回复的特别标记证书。OCSP 证书的密钥扩展项与一般证书不一致。

一个确定的回复信息由以下组成：

——回复语法的版本

——OCSP 服务器名称

——对每一张被提及证书的回复

——可选扩展

——签名算法对象标识符号

——对回复信息杂凑后的签名

对每一张被请求证书的回复包括

——目标证书识别

——证书状态值

——回复有效期

——可选扩展

这个说明定义了以下在证书状态值中使用的一些确定回复识别：

——良好

——已撤消

——未知

“良好”状态表示一个对状态查询的积极回复。至少，这个积极回复表示这张证书没有被撤消，但是不一定意味着这张证书曾经被颁发过或者产生这个回复在证书有效期内。回复扩展可以被用来传输一些附加信息，服务器由此可以对这张证书的状态做出一些积极的声明，诸如（已颁发）保证，有效期等等。

“已撤消”状态表示证书已被撤消（无论是临时性的还是永久性的（待判断），证书冻结或挂起也是这种状态）

“未知”状态表示服务器不知道请求的证书，可能不是本 CA 签发。

万一出错，OCSP 服务器会返回一个出错信息。

这些信息无须签名。出错信息可以是以下一些类型

——未正确格式化的请求（malformedRequest）

——内部错误（internalError）

——请稍后再试(trylater)

——需要签名(sigRequired)

——未授权(unauthorized)

如果接受到一个没有遵循 OCSP 语法的请求，服务器产生“未正确格式化的请求”回复。

回复“内部错误”表示 OCSP 服务器处于一个不协调的内部状态。请求需要再试，暗示尝试另一个服务器。

如果 OCSP 服务器正在工作，但是不能返回被请求证书的状态，那么“稍后再试”回复能被用来表示服务存在，但暂时不能响应。

当服务器需要客户端签名请求后才能产生一个正确回复时，将回复“需要签名”。

当客户端未被授权允许向这台服务器发送请求时，回复“未授权”将被返回。一般其它 CA 用户的访问都被认为未授权，除非有协议或支持交叉认证的系统，有的系统即使支持交叉认证，对没有签发交叉认证证书的 CA 用户也视为未授权。

回复信息可以在其中包含三种时间——此次更新，下次更新和产生时间。这些域的语义如下：

——此次更新：此证书状态被表示为正确的时间

——下次更新：在此时间之后，可获得此证书状态的新近信息

——产生时间：OCSP 签名这个回复的时间

如果服务器未设置下次更新，那意味着证书状态随时可以更新，用户得到的只是目前证书的状态。

如果一个 OCSP 服务器知道一个特定的 CA 私钥不安全，那么针对所有这个 CA 颁布的证书都可以返回一个撤消状态。

为了传达给 OCSP 客户端一个知道的信息获取点，CA 可以在证书内注明 OCSP 访问点，同时也可以在 OCSP 客户端本地配置 OCSP 提供者获取地（信息）。

支持 OCSP 服务的 CA，无论是自身实现还是通过授权服务器来提供，都必须提供包括统一资源识别形式的获取地信息和在一个获取描述序列中的对象识别符号形式的获取方法。

在主体证书的获取地域中的值定义了使用什么传输（例如 HTTP）来获取 OCSP 服务器并且可以包含其他传输相关信息（例如 URL）。

在接受一个已签名的回复为有效之前，OCSP 客户端必须确认：

在回复信息中所指的证书和相应请求中所指证书一致。

回复中的签名有效。

签名者的身份和相应接受请求者匹配。

签名者正被授权签名回复。

表示状态被认为是正确的时间（此次更新）足够新。

如果有的话，下次更新的时间应该晚于现时时间。

这里的ASN.1语法引用了在GB/T 20518－2006中定义的术语。至于签名运算,被签名的数据使用ASN.1显示编码规则（DER）[x.690]。

除非指定了其他，否则默认使用ASN.1外在标记。从别处引用的的术语有：扩展(Extensions)，证书序列号（CertificateSerialNumber）,主体公钥信息(SubjectPublicKeyInfo)，名称(Name)，算法识别(AlgorithmIdentifier)，证书撤消列表原因（CRLReason）

```
OCSPRequest::= SEQUENCE {
        tbsRequest                  TBSRequest,
        optionalSignature   [0]  EXPLICIT Signature OPTIONAL }
TBSRequest::= SEQUENCE {
        version             [0]  EXPLICIT Version DEFAULT v1,
        requestorName       [1]  EXPLICIT GeneralName OPTIONAL,
        requestList              SEQUENCE OF Request,
        requestExtensions   [2]  EXPLICIT Extensions OPTIONAL }
Signature::= SEQUENCE {
        signatureAlgorithm       AlgorithmIdentifier,
        signature                BIT STRING,
        certs                [0] EXPLICIT SEQUENCE OF Certificate   OPTIONAL
}
Version::= INTEGER  {  v1(0) }
Request::= SEQUENCE {
        reqCert                    CertID,
        singleRequestExtensions    [0] EXPLICIT Extensions OPTIONAL
}
CertID::= SEQUENCE {
        hashAlgorithm       AlgorithmIdentifier,
        issuerNameHash      OCTET STRING, -- Hash of Issuer's DN
        issuerKeyHash       OCTET STRING, -- Hash of Issuers public key
        serialNumber        CertificateSerialNumber }
```

发布者名称杂凑(issuerNameHash)是发布者甄别名称的杂凑值。应该检测证书发布者名称域的DER编码的杂凑值。发布者密钥杂凑（issuerKeyHash）是发布者公钥的杂凑值。对发布者证书的主体公钥域的值（不包括标签和长度）进行杂凑。所有这些使用的杂凑算法都由杂凑算法域(hashAlgorithm)确定。序列号域（serialNumber）是被查询状态证书的序列号。

除了对CA名称进行杂凑外还对CA的公钥进行杂凑，这样做的主要原因是为了识别发布者，因为两个CA有可能选择同一名称(虽然建议使用独一无二的名称，但这并不是强制要求的)。然而，除非两个CA明确表示他们共享私钥或者其中一个CA的密钥是不安全的，否则两个CA是不可能有相同密钥的。

支持任何的扩展是可选的。每一个可选扩展的关键性标志都不应该被设置。未被承认的扩展必须被忽略。服务器可以选择签名OCSP的请求。在那种情况下，签名是根据TBS请求(tbsRequest)结构计算得到的。

一个OCSP服务回复至少包括用来指示对先前请求所处理状态的回复状态域（responseStatus)。如果回复状态域的值是某一种出错情况，那么回复字节(responseBytes)将不被设置。

```
OCSPResponse ::= SEQUENCE {
```

```
      responseStatus             OCSPResponseStatus,
      responseBytes          [0] EXPLICIT ResponseBytes OPTIONAL
 }
OCSPResponseStatus ::= ENUMERATED {
       successful            (0),  --Response has valid confirmations
       malformedRequest      (1),  --Illegal confirmation request
       internalError         (2),  --Internal error in issuer
       tryLater              (3),  --Try again later
                                   --(4) is not used
       sigRequired           (5),  --Must sign the request
       unauthorized          (6)   --Request unauthorized
   }
```

回复字节（responseBytes）的值由对象标识和一个编码成 OCTET 字符串的回复标记（此标记由刚才的对象标识确定）组成。

```
ResponseBytes ::= SEQUENCE {
       responseType   OBJECT IDENTIFIER,
       response       OCTET STRING
 }
```

对于基本的 OCSP 回复，回复类型是 id-pkix-ocsp-basic。

```
id-pkix-ocsp           OBJECT IDENTIFIER ::= { id-ad-ocsp }
id-pkix-ocsp-basic     OBJECT IDENTIFIER ::= { id-pkix-ocsp 1 }
```

OCSP 服务响应应该有能力产生 id-pkix-ocsp-basic 回复类型的回复。同样的，OCSP 客户端也应该有能力接受并且处理 id-pkix-ocsp-basic 类型的回复。

回复的值应该是基本 OCSP 回复（BasicOCSPResponse）的 DER 编码。

```
BasicOCSPResponse ::= SEQUENCE {
      tbsResponseData      ResponseData,
      signatureAlgorithm   AlgorithmIdentifier,
      signature            BIT STRING,
      certs            [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}
```

签名值应该对回复数据（ResponseData）的 DER 编码上的杂凑计算而得。

```
ResponseData ::= SEQUENCE {
      version              [0] EXPLICIT Version DEFAULT v1,
      responderID              ResponderID,
      producedAt               GeneralizedTime,
      responses                SEQUENCE OF SingleResponse,
      responseExtensions   [1] EXPLICIT Extensions OPTIONAL }
ResponderID ::= CHOICE {
      byName               [1] Name,
      byKey                [2] KeyHash
}
KeyHash ::= OCTET STRING -- SHA-1 hash of responder's public key
   (不包括标签和长度域)
SingleResponse ::= SEQUENCE {
```

```
        certID                      CertID,
        certStatus                  CertStatus,
        thisUpdate                  GeneralizedTime,
        nextUpdate              [0] EXPLICIT GeneralizedTime OPTIONAL,
        singleExtensions        [1] EXPLICIT Extensions OPTIONAL
    }
     CertStatus ::= CHOICE {
           good                  [0] IMPLICIT NULL,
           revoked               [1] IMPLICIT RevokedInfo,
           unknown               [2] IMPLICIT UnknownInfo
    }
    RevokedInfo ::= SEQUENCE {
           revocationTime              GeneralizedTime,
           revocationReason        [0] EXPLICIT CRLReason OPTIONAL
     }
    UnknownInfo ::= NULL——这个可以被一个列举代替。
```

下面简单列出一些注意事项：

此次更新和下次更新域定义了一个推荐的有效期。一个时间长度和证书撤消列表中的时间长度相一致。如果下次更新的值早于当前本地系统时间，那么这个回复将被认为不可靠。如果此次更新的值晚于当前本地系统时间，那么这个回复也将被认为不可靠。产生时间是这个回复被签名的时间。

用来签名证书状态信息的密钥可以和签名证书状态的密钥不同。但是必须保证签名这个信息的实体已被授权。所以证书发布者必须自己签名 OCSP 回复或者明确指派这个权利给其他实体。OCSP 签名代表可以通过包含在 OCSP 回复签名者证书扩展密钥用途扩展中的 id-kp-OCSPSigning 来指派。这张证书必须直接由颁布所涉及证书的 CA 发布。

```
id-kp-OCSPSigning OBJECT IDENTIFIER ::= {id-kp 9}
```

依赖 OCSP 回复的系统和应用程序必须有能力探测并且执行 id-ad-ocspSigning 值的使用，如前所述。他们可以提供一种本地配置一个或更多个 OCSP 签名权威机构的方法，而且可以指定一组被信任的签名权威机构。当要求验证回复上签名的证书未满足以下其中一个标准时，他们必须拒绝这样的回复，必须满足的具体标准是：

和本地配置的对所涉及证书的 OCSP 签名权威机构匹配；

或者和颁发所涉及证书的 CA 相同；

或者包括在扩展密钥用途扩展中的 id-ad-ocspSigning 值，这种证书由颁发所涉及证书的 CA 颁发。

回复本身可以用来验证回复中签名的证书，也可以应用于其他接受或者拒绝的标准。

利用随机数绑定了请求和回复，用来防止重发攻击（replay attacks)。随机数作为一个请求扩展被包括在请求中，同样的也作为一个回复扩展被包括在回复中。在请求和回复中，随机数由对象标识 id-pkix-ocsp-nonce 识别，其中 extnValue 包含了随机数的值。

```
id-pkix-ocsp-nonce      OBJECT IDENTIFIER ::= { id-pkix-ocsp 2 }
```

一个 OCSP 客户端可以希望指定一个它所理解的回复类型。为了达到这样的目的，它应该使用 id-pkix-ocsp-response 对象标识符的扩展，并且值为可接受回复（AcceptableResponses)。

这个扩展作为一个请求扩展被包括在请求中。在可接受回复中包括了这个客户端可接受的不同回复类型的对象标识符号（例如，id-pkix-ocsp-basic)。

```
id-pkix-ocsp-response  OBJECT IDENTIFIER ::= { id-pkix-ocsp 4 }
AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER
```

OCSP 服务器要有能力回复一个 id-pkix-ocsp-basic 的回复类型。OCSP 客户端也应该有能力接受并处

理 id-pkix-ocsp-basic 回复类型的回复。

5.6.2 SOCSP 证书状态查询协议

SOCSP 协议是根据我国实际应用特点自主研制的协议标准。协议格式同 OCSP 协议，只是签名不再需要，实现时应考虑使用 HMAC 代替签名。

6 协议报文语法

6.1 加密数据报文

本规范中加密数据报文为不附带任何编码标示的原始密文数据。

6.2 摘要数据报文

Digestinfo：=SEQUENEC{
digestAlgorithm DigestAlgorithmidentifier,
digest Digest}

DigestInfo 类型的各个字段有如下含义：

——DigestAlgorithmIdentifier ::= AlgorithmIdentifier
类型识别消息摘要算法。包括 SM3、SHA_1、SHA_256。消息摘要算法把任意长度的八位字节串（消息）映射成定长的字节串（消息摘要）。

——Digest::=OCTET STRING
是消息摘要进程的结果。

6.3 数字签名报文

SignedData ::= SEQUENCE {
version [1] Version DEFAULT v1997,
digestAlgorithms DigestAlgorithmIdentifiers,
contentInfo ContentInfo,
certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
signerInfos SignerInfos }

SignedData 类型的各个字段有如下含义：

——Version ::= INTEGER
语法版本号，对于本标准的这个版本为 1。

——DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
消息摘要算法标识符。集合中可能有任意多的码元，包括 0 在内。每个码元标识消息分类算法（和任意相关的参数），内容通过这个算法为某个签名者摘要。该集合用来按任意顺序列出所有签名者使用的消息摘要算法，以简化单遍签名验证。

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

——是已经签名的内容。它可以有任意个定义内容类型。

——certificates 是一套 PKCS #6 扩展证书和 X.509 证书。

——ExtendedCertificatesAndCertificates ::=
SET OF ExtendedCertificateOrCertificate

——ExtendedCertificateOrCertificate ::= CHOICE {certificate Certificate,-- X.509
extendedCertificate [0] IMPLICIT ExtendedCertificate}

——crls 是一组 GB/T 20518－2006 证书撤消名单。

——SignerInfos ::= SET OF SignerInfo

SignerInfo ::= SEQUENCE {

version Version,

issuerAndSerialNumber IssuerAndSerialNumber,

digestAlgorithm DigestAlgorithmIdentifier,

authenticatedAttributes [0] IMPLICIT Attributes OPTIONAL,

digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,

encryptedDigest EncryptedDigest,

unauthenticatedAttributes [1] IMPLICIT Attributes OPTIONAL }

SignerInfo 类型的各个字段有如下含义：

——Version ::= INTEGER

语法版本号，对于本标准的这个版本为 1。

——IssuerAndSerialNumber::= INTEGER

签名者公钥证书 CertificateSerialNumber。

——DigestAlgorithmIdentifier::= AlgorithmIdentifier

消息摘要算法标识（和任意相关参数）。在这种算法下，内容和鉴别属性（如果存在的话）被摘要。它应当在超 SignerInfo 值的 digestAlgorithms 字段中。

——authenticatedAttributes::=OCTET STRING

是一组由签名者签名（既鉴别）的属性，字段是可任选的。如果被签名的 ContentInfo 值的内容类型不是数据，字段一定存在。如果字段存在，它至少要包含两个属性：

——被签名 ContentInfo 值的内容；

——内容的消息摘要。

—— 在这里可能有用的其它属性类型，如签名时间，也在 PKCS#9 中定义。

——digestEncryptionAlgorithm ::= AlgorithmIdentifier

标识摘要加密算法。通过这种算法，消息摘要和相关信息由签名者的私有密钥加密。

——encryptedDigest ::= OCTET STRING

是用签名者私有密钥加密的消息分类和相关信息的结果。

——unauthenticatedAttributes::= OCTET STRING

是一组签名者没有签名（既鉴别）的属性。字段是可选择的。在这里可能有用的属性类型（如连署签名）在 PKCS#9 中定义。

6.4 数字信封报文

SignedAndEnvelopedData ::= SEQUENCE {

version Version,

recipientInfos RecipientInfos,

digestAlgorithms DigestAlgorithmIdentifiers,

encryptedContentInfo EncryptedContentInfo,

certificates [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,

crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,

signerInfos SignerInfos }

SignedAndEnvelopedData 类型的各个字段有如下含义：

——Version ::= INTEGER

语法版本号。对于本标准的这个版本，它为 1。

——RecipientInfos

```
RecipientInfo ::= SEQUENCE {
  version Version,
  issuerAndSerialNumber     IssuerAndSerialNumber,
  keyEncryptionAlgorithm    KeyEncryptionAlgorithmIdentifier,
  encryptedKey EncryptedKey }
```

- Version ::= INTEGER

语法版本号。对于本标准的这个版本，它为0。

a) IssuerAndSerialNumber::= INTEGER

签名者公钥证书CertificateSerialNumber。

b) KeyEncryptionAlgorithmIdentifier::= AlgorithmIdentifier

密钥加密算法标识（和任意相关参数），内容加密密钥由接收者的公共密钥加密。

c) EncryptedKey ::= OCTET STRING

接收者的公共密钥加密内容加密密钥的结果

——DigestAlgorithmIdentifiers::= AlgorithmIdentifier

标识摘要加密算法。通过这种算法，消息摘要和相关信息由签名者的私有密钥加密。

——EncryptedContentInfo

```
EncryptedContentInfo ::= SEQUENCE {
  contentType                   ContentType,
  contentEncryptionAlgorithm    ContentEncryptionAlgorithmIdentifier,
  encryptedContent              [0] IMPLICIT EncryptedContent OPTIONAL }
```

a) ContentType ::= OBJECT IDENTIFIER

表示内容的类型，是一个对象标识符

b) ContentEncryptionAlgorithmIdentifier::= AlgorithmIdentifier

标识一个内容加密算法，内容加密算法支持加密和解密操作。

c) encryptedContent::= OCTET STRING

加密该内容的结果。字段是可选择的，如果字段不存在，其扩展值必须由其它方式提供。

——certificates是一套PKCS #6扩展证书和GB/T 20518—2006证书。

```
ExtendedCertificatesAndCertificates ::=
SET OF ExtendedCertificateOrCertificate

ExtendedCertificateOrCertificate ::= CHOICE {certificate Certificate,-- X.509
extendedCertificate          [0] IMPLICIT ExtendedCertificate}
```

——crls是一组GB/T 20518—2006证书撤消名单。

——SignerInfos ::= SET OF SignerInfo。

附 录 A
（规范性附录）
系统与格式定义

A.1 证书格式

本文档中所描述证书，其格式应符合 GB/T 20518-2006《信息安全技术 公钥基础设施 数字证书格式》。

不同的 PKI 管理消息要求消息的发起者指明证书中必备的一些域。证书模板结构允许一个终端实体或者 RA 尽其可能指定其要求的证书内容。除了域内容是随意填写之外，证书模板可以等同于一个证书。

注意到即使消息的发起者指定全部它所需求的证书内容，而 CA 在签发证书时有权进行修改。如果这样签发的证书不能被消息请求者接受，导致的后果可能拒绝确认消息，或者发送一个错误消息（同时携带 PKI 状态码——“拒绝”）。

证书模板语法说明如下：

```
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
}
```

A.2 证书撤销列表CRL格式

本文档中所描述证书撤销列表 CRL，其格式应符合 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》。

A.3 加密值

当在 PKI 消息中发送加密值时（具体地，在本文档中指私钥或者证书），需要使用加密数据结构来组织。

使用该数据结构时要求发送者和期望的接收者都能进行加解密运算。本规范推荐发送者和接收者进行加解密运算，或者能够产生一个共享密钥。

如果 PKI 消息的接收者已经拥有一个用来解密的私钥，那么“encSymmKey“域可能包含一个使用接收者公钥加密的会话密钥。

加密值的语法说明如下：

```
EncryptedValue ::= SEQUENCE {
intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
-- 加密值期望使用的算法
symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
```

```
-- 用于加密数据的对称算法类型
encSymmKey    [2] BIT STRING           OPTIONAL,
-- 加密后的对称密钥
keyAlg        [3] AlgorithmIdentifier  OPTIONAL,
-- 用于加密对称密钥的算法类型
valueHint     [4] OCTET STRING         OPTIONAL,
-- 关于加密内容的简短说明
encValue       BIT STRING
-- 对称加密结果
}
```

A.4 PKI消息的状态码和故障信息

所有的响应消息都包含一些状态信息，下面定义了一些状态码

```
PKIStatus ::= INTEGER {
granted                 (0),
-- 按照请求准确地返回结果
grantedWithMods         (1),
-- 按照请求返回了一些近似的结果，请求者负责确定其中的区别
rejection               (2),
-- 拒绝请求，在消息其他地方将有更多的信息
waiting                 (3),
-- 等待，请求尚没有被处理
revocationWarning       (4),
-- 撤销警告，该消息警告了一个即将来临的撤销
revocationNotification (5),
-- 撤销通知，通知撤销已经发生
keyUpdateWarning        (6)
-- 密钥更新请求消息中指定的密钥已经更新
}
```

响应者可以使用下面的语法来提供更多的关于故障的信息。

```
PKIFailureInfo ::= BIT STRING {
-- 在需要的时候，将来可以添加更多的错误码
badAlg           (0),
-- 不能识别或者不支持的算法标识符
badMessageCheck  (1),
-- 消息完整性检查失败（比如签名未能验证通过）
badRequest       (2),
-- 不允许或者不支持该事务
badTime          (3),
-- 根据本地策略，消息时间和系统时间差别较大
badCertId        (4),
-- 没有找到合适的证书
badDataFormat    (5),
```

```
-- 提交的数据格式错误
wrongAuthority   (6),
-- 请求中指定的权威和创建响应者令牌的权威不相同
incorrectData    (7),
-- 请求者数据不正确（用在公正服务中）
missingTimeStamp (8),
-- 在需要的地方（根据策略）缺失时间戳
badPOP           (9)
-- 拥有证明结构错误
}
PKIStatusInfo ::= SEQUENCE {
status        PKIStatus,
statusString  PKIFreeText     OPTIONAL,
failInfo      PKIFailureInfo  OPTIONAL
}
```

A.5 证书识别

为识别特定的证书，需要用到证书 ID（certID）数据结构，CertID 的语法说明如下：

```
CertID ::= SEQUENCE {
    hashAlgorithm           AlgorithmIdentifier,
    issuerNameHash          OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash           OCTET STRING, -- Hash of Issuers public key
    serialNumber            CertificateSerialNumber }
```

A.6 带外根CA公钥

每个根 CA 必须将其当前公钥通过带外的方式发布，尽管关于发布机制的讨论超出了本文档的范围，定义了支持这些机制的数据结构。

一般地，现在有两种方法发布根 CA 公钥：一种是 CA 直接发布其自签名证书；另外一种是通过目录服务发布，而且 CA 同时发布该公钥的 HASH 值，以便允许验证公钥的完整性。

```
OOBCert ::= Certificate
```

该证书中的域有如下限制：

——证书必须是自签名的（也就是说，签名必须能够使用证书中公钥信息来验证）；

——主题域（subject）和签发者域（issuer）必须相同；

——如果主题域（subject）为空，那么主题可选名称（subjectAltNames）和签发者可选名称（issuerAltNames）必须填写，而且两者完全相同；

——其他扩展项的值必须符合自签名证书要求（比如，主题密钥标识和签发者密钥标识必须相同）。

```
OOBCertHash ::= SEQUENCE {
hashAlg         [0] AlgorithmIdentifier     OPTIONAL,
certId          [1] CertId                  OPTIONAL,
hashVal         BIT STRING
}
```

使用 HASH 值的目的是使任何可以安全得到该 HASH 值的用户（通过带外的方式得到）都可以验证该 CA 的自签名证书。

A.7 存档选项

请求者在需要时可以使用”PKIArchiveOptions“存档选项，用来表明他们要求 PKI 存档私钥。

存档选项 PKIArchiveOptions 语法说明如下：

```
PKIArchiveOptions ::= CHOICE {
encryptedPrivKey        [0] EncryptedKey,
-- 私钥
keyGenParameters        [1] KeyGenParameters,
-- 允许重新产生私钥的参数
archiveRemGenPrivKey    [2] BOOLEAN
-- 如果发送者期望接受者归档其对应请求所生成密钥对中的私钥，
-- 则设置为真，如果不需要归档，则设置为假。
}
```

A.8 发布信息

请求者可以使用”PKIPublicationInfo“选项来表明他们希望 PKI 发布证书。发布信息 PKIPublicationInfo 语法说明如下：

```
PKIPublicationInfo ::= SEQUENCE {
action     INTEGER {
dontPublish (0),
pleasePublish (1) },
pubInfos  SEQUENCE SIZE (1..MAX) OF SinglePubInfo OPTIONAL
}
```

如果 action 项为"dontPublish"（不发布），则 pubInfos 必须为空。

附 录 B
（资料性附录）
RA 与 CA 间相关协议

B.1 RA的服务模式

RA 服务器通常有两种服务模式，一种是 C/S 模式，一种是 B/S 模式。目前，RA 服务器大都以 Web 方式提供服务。RA 服务器的程序大致可发分为两部分：前台页面程序和后台服务程序。前台页面程序主要用于与客户端进行交互，通过页面表单方式实现客户端与 RA 服务器的信息交换，而客户端利用 IE 浏览器下载并执行前台页面程序。后台服务程序在 RA 服务器上运行，用来实现 RA 内在的业务流程，并与 CA 服务器进行通信，实现证书的签发、撤消、更新、恢复等功能。

B.2 RA前台页面程序

RA 前台页面程序用来与客户端进行交互，客户端通常为 Windows 操作系统平台，利用 IE 浏览器下载并执行前台页面程序。客户端通常采用 ActiveX 技术提供证书载体服务。

RA 前台页面程序主要包括注册页面、审核页面和执行页面。

注册页面主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复注册等；

审核页面主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复审核等；

执行页面主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复执行等。

ActiveX 提供的证书载体服务主要包括设备管理类接口、密码服务类接口和证书管理类接口。

设备类接口主要包括打开设备、关闭设备、格式化设备、获取设备信息、创建容器、删除容器、获取容器信息、设置（修改）Pin 码、验证 Pin 码；

密码服务类接口主要包括摘要、随机数生成、对称加密、对称解密、密钥对生成（非对称密钥）、密钥对导入、公钥导出、数字签名、签名验证、非对称加密、非对称解密；

证书管理类接口主要包括导入证书、导出证书、验证证书、解析证书（取序列号、公钥、DN 及扩展项）。

B.3 RA后台服务程序

RA 服务器利用 Web 程序（jsp、asp、c#等）提供后台服务，实现 RA 内在业务流程并与 CA 进行通信，主要包括三类服务：注册服务、审核服务和执行服务。

注册服务主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复注册等；

审核服务主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复审核等；

执行服务主要包括证书申请、撤销、更新、冻结、解冻、密钥恢复执行等。

RA 与 CA 间的通信协议参照 PKIX-证书管理协议。本节所有用于 PKI 管理的消息都采用下列结构：

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

PKIHeader　　消息头即各种 PKI 消息共有的信息。

PKIBody　　消息体即与具体消息相关的信息。

PKIProtection　　用于保护 PKI 消息的比特串(保护消息的完整性)。

extraCerts 可以包含对接收者有用的证书。例如, CA 或 RA 可以利用该字段为用户提供用于验证用户新证书的证书。注意，该字段中包含的不一定是证书链 ，接收者要使用这些证书的话，可能需要对这些证书进行排序、选择或处理。

PKI Message Header 采用下面的数据结构：

PKIHeader ::= SEQUENCE {

pvno INTEGER { ietf-version2 (1) },

sender GeneralName, 标识发送者

recipient GeneralName, 标识预期的接收者

messageTime [0] GeneralizedTime OPTIONAL,

产生本消息的时间(当发送者认为接收到该消息时这个时间仍有意义的情况下使用)

protectionAlg [1] AlgorithmIdentifier OPTIONAL,计算保护比特的算法

senderKID [2] KeyIdentifier OPTIONAL,

recipKID [3] KeyIdentifier OPTIONAL,标识用于保护的特定密钥

transactionID [4] OCTET STRING OPTIONAL,标识 transaction； 在相关的请求、响应、确认消息中本字段值相同

senderNonce [5] OCTET STRING OPTIONAL,

recipNonce [6] OCTET STRING OPTIONAL,

用于提供重放保护的随机数，senderNonce 由该消息的产生者插入；

recipNonce 是本消息预期接收者之前在相关消息中插入的随机数

freeText [7] PKIFreeText OPTIONAL,用于上下文相关的人可读说明

generalInfo [8] SEQUENCE SIZE (1..MAX) OF InfoTypeAndValue OPTIONAL 用于上下文相关的可读信息

}

PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String

pvno CMP 的当前版本号，取固定值 1

sender PKIMessage 发送者的名字。该名字与 senderKID(提供的话)一起应当可以用于验证对该消息的保护。如果发送方实体不知道任何有关 sender 的信息(如，在初始化请求消息中，EE 可能不知道自己的 DN、e-mail name, IP address 等)，那么"sender"字段必须包含值"NULL"；即编码为长度为 0 的 SEQUENCE OF RDN。在这种情况下，senderKID 必须包含一个标识符(即 reference number)通知接收者用于验证消息的相关的共享秘密信息。

recipient PKIMessage 接收者的名字。该名字与 recipKID(提供的话)一起应可以用于验证对该消息的保护。

protectionAlg 指定用于保护消息的算法。如果未提供用作保护的比特位(注意 KIProtection 可选)，则该字段必须省略；若提供用作保护的比特位，则必须提供该字段。

senderKID, recipKID 用于指明使用哪一密钥对消息进行保护(通常，仅当使用 DH 密钥保护消息时才要求 recipKID 出现)。

transactionID 该字段允许响应消息的接收者将响应与先前发送的请求相互关联起来。例如，当 RA 在给定时刻有多个正在处理的请求的情况下。

senderNonce, recipNonce 保护 PKIMessage 免受重放攻击。

messageTime 包含发送者产生该消息的时间。可由 EE 用于根据中央系统的时间调整自己的本地时间。

freeText 可用于向接收者发送人可读的消息(使用任意多种语言)。在这个序列中使用的第一种语言指明期望响应所用的语言。

generalInfo 可用于向接收者发送机器可读的附加数据。

PKIBody ::= CHOICE {与消息相关的体元素

```
ir       [0]  CertReqMessages,         Initialization Request
ip       [1]  CertRepMessage,          Initialization Response
cr       [2]  CertReqMessages,         Certification Request
cp       [3]  CertRepMessage,          Certification Response
p10cr    [4]  CertificationRequest,    PKCS#10 certification request
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
}
```

某些 PKI 消息需要保护其完整性。(注意，如果使用非对称算法保护消息且相关公钥部分已被认证，那么消息的来源也可以被认证。另一方面，如果公钥部分未被认证，那么消息来源不能够被自动认证，但可以通过外部系统方法进行认证)。

对 PKI 消息实行保护时，采用下面的结构：

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

计算 PKIProtection 所用的输入是下列数据结构 DER 编码：

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

在有些情况下，可以不使用 PKIProtection 比特串来保护消息(即省略该可选项)，因为可以使用 PKIX 之外的其它保护方法。这种外部保护方法的例子包括 PKIMessage 的 PKCS #7 和 Security Multiparts[RFC1847]封装(或者仅仅是 PKIBody(省略 CHOICE tag) - 如果相关的 PKIHeader 信息由外部机制安全地传送)；利用 PKCS #7 进行保护的规范在另外的文档(CMM over CMS)中给出。然而应当注意的是，许多这样的外部机制要求用户已经拥有公钥证书和/或唯一的 DN 和/或其它类似的信息。因此，这些方法不适用于初始化注册、密钥恢复或其它的具有启动特性的过程。在这些情形下，必须使用可选的 PKIProtection 以保护其完整性。

下面介绍几个通用消息：

a) 正确确认消息（PKI Confirmation content）：

这一数据结构在证书管理协议中作为最后的 PKIMessage。在所有情况下它的内容都相同，即没有任何内容，因为 PKIHeader 中包含了所有要求的信息。

PKIConfirmContent ::= NULL

b) 错误确认消息（Error Message content）

如果对请求的处理发生错误，则可能返回错误消息。其语法定义如下：

```
ErrorMsgContent ::= SEQUENCE {
pKIStatusInfo   PKIStatusInfo,
errorCode       INTEGER     OPTIONAL,    与实现相关的错误码
errorDetails    PKIFreeText OPTIONAL     与实现相关的错误细节描述
}
```

c) 所有的响应消息都包含一些状态信息。定义了下列状态值。

```
PKIStatusInfo ::= SEQUENCE {
status          PKIStatus,
statusString    PKIFreeText     OPTIONAL,
failInfo        PKIFailureInfo  OPTIONAL
}
PKIStatus ::= INTEGER {
granted         (0),you got exactly what you asked for;
grantedWithMods (1),you got something like what you asked for; he 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 the key update request message
}
```

d) 在错误的情形下,响应者可以使用下面的语法来提供更多的信息。

```
PKIFailureInfo ::= BIT STRING {
badAlg      (0),unrecognized or unsupported Algorithm Identifier
badMessageCheck (1),integrity check failed(eg,signature didn't 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  (used for notary(公证人) services)
missingTimeStamp(8),when the timestamp is missing but should be there (by policy)
badPOP          (9) the proof-of-possession failed
}
```

B.4 证书申请协议

正常流程如下图分为以下步骤完成，如果 CA 收到请求后发现错误或其它原因不能正确回应，CA 将直接返回错误消息，并中断此次连接。

```
        RA                                              CA
================                                ================
                                                  带外分发 RA、CA 证书
  产生认证请求
                --->>--请求随机数 recipNonce-->>--
                                                    产生 recipNonce 响应
                --<<--接收随机数 recipNonce--<<--

  产生密钥
  产生发送随机数 senderNonce
  填写申请内容
  利用签名保护请求
                  -->>--证书申请-->>--
                                                    验证请求
                                                      比较 recipNonce
                                                    处理请求
                                                    产生响应
                 --<<--证书回应--<<--

  处理响应
  比较 senderNonce
  产生确认消息
                  -->>--确认消息-->>--
                                                  验证确认消息
                                                      比较 recipNonce
```

（当验证确认消息失败时，如果新签发的证书已经发布或可通过其它方式获得，那么 CA 必须作废这个证书。）

a) 申请结构内容填写

PKIBody 为 CertReqMessages, 命令字是 cr, 其中指定了请求的 certificate(s)。通常情况下，对每一个请求都提供 SubjectPublicKeyInfo，KeyId 和 Validity 字段。这一消息用于实体初次与 PKI 进行交互。

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

其中，regInfo 应当仅包含与认证请求上下文相关的辅助信息，如果完成认证请求需要这些信息的话。这些信息可以包括用户的联系信息、计费信息或其它有助于完成认证请求的辅助信息。

直接与证书内容相关的信息应当包含在 certReq 中。然而，如果 RAs 在 certReq 中包含额外的内容可能使 pop 字段无效。因此，RA 计划放入证书内容中的信息可以放入 regInfo。CRMF 中定义的注册信息包括: id-regInfo-asciiPairs 和 id-regInfo-certReq。

CertRequest 中可能包含一个或多个与处理请求相关的控件。CRMF 中定义的控件包括：regToken、authenticator、pkiPublicationInfo、pkiArchiveOptions、oldCertID 及 protocolEncrKey。

许多 PKI 管理操作要求消息的发送方指明需要在证书中出现的字段，如 ir、cr、kur、ccr 等。CertTemplate 结构允许 RA 为其申请的证书指定任意多的字段。CertTemplate 与 Certificate 结构相同，但所有的字段都为可选字段。

注意，即使发送方为申请的证书指定了所有内容，CA 仍可在实际签发的证书中任意地修改各个字段。如果请求者不接受被修改的证书,可以拒发 Confirmation 消息,或发送一个 Error 消息(其中的 PKIStatus 为"rejection")。

```
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
}
```

b) 申请回复

PKIBody 为 CertRepMessage，命令字是 cp,对所请求的每一个证书，CertRepMessage 中都包含一个 PKIStatusInfo 字段、一个申请者证书、还可能有一个加密的私钥(通常由一个会话密钥加密，而会话密钥本身由 protocolEncKey 加密)。

PKIBody 为 CertRepMessage，对每一个请求，CertRepMessage 中都包含一个状态，可选的 CA 公钥、失败信息、申请者证书和加密的私钥。

```
CertRepMessage ::= SEQUENCE {
caPubs              [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL,
response                SEQUENCE OF CertResponse
}
CertResponse ::= SEQUENCE {
certReqId    INTEGER,将该响应与相应的请求进行匹配(请求中未指定 certReqId 值时，取值为-1)
status                  PKIStatusInfo,
certifiedKeyPair        CertifiedKeyPair    OPTIONAL,
rspInfo                 OCTET STRING        OPTIONAL
```

与为[CRMF]中 CertReqMsg 的 regInfo 字段定义的 id-regInfo-asciiPairs 相似

}
CertifiedKeyPair ::= SEQUENCE {
certOrEncCert CertOrEncCert,
privateKey [0] EncryptedValue OPTIONAL,
publicationInfo [1] PKIPublicationInfo OPTIONAL
}
CertOrEncCert ::= CHOICE {
certificate [0] Certificate,
encryptedCert [1] EncryptedValue
}

依赖于响应的状态,在每一个 CertResponse 中 PKIStatusInfo 的 failInfo 和 CertifiedKeyPair 中的证书只能出现一个。对某些状态值(如 waiting),任何可选的字段都不会出现。

给定了 EncryptedCert 和相关的解密密钥就可以获得证书。这样做的目的是允许 CA 返回证书, 但进行限制使得只有预期的接收者才能获得实际的证书。采用这种方法,即使没有证据能够证明请求者就是可以使用相关私钥的 EE,CA 也可以为其返回携带证书的响应(注意,直到 CA 接收到 PKIConfirm 消息,才能够获得证据)。因此,CA 不必在 POP 发生错误的情况下作废该证书。

若在 PKI 消息中要发送加密的值(CMP 中限制为私钥或证书),使用 EncryptedValue 类型的数据结构。

EncryptedValue ::= SEQUENCE {
intendedAlg [0] AlgorithmIdentifier OPTIONAL, the intended algorithm for which the value will be used
symmAlg [1] AlgorithmIdentifier OPTIONAL,用于加密值的对称算法
encSymmKey [2] BIT STRING OPTIONAL,加密后的用于加密值的对称密钥
keyAlg [3] AlgorithmIdentifier OPTIONAL,用于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(仅对发送方有意义,且仅当发送方将来需要检查 EncryptedValue 才可能使用)
encValue BIT STRING
}

使用这一数据结构要求产生者和预期的接收者都能够进行加密和解密。这通常意味着发送者和接收者有或可以产生共享密钥。

B.5 证书撤销协议

RA 请求作废一个或几个证书时,使用下面的数据结构。RA 的名字在 PKIHeader 结构中。流程参照证书申请过程

申请的 PKIBody 为 RevReqContent,命令字是 rr。

RevReqContent ::= SEQUENCE OF RevDetails
RevDetails ::= SEQUENCE {
certDetails CertTemplate,--允许请求者为请求作废的证书指定尽可能多的信息(如:在 serialNumber 未知的情况下)
revocationReason ReasonFlags OPTIONAL,请求作废的原因 crlEntryExtensions
badSinceDate GeneralizedTime OPTIONAL,
crlEntryDetails Extensions OPTIONAL 请求的 crlEntryExtensions
}

CA 响应的 PKIBody 为 RevRepContent,命令字是 rp。

RevRepContent ::= SEQUENCE {

status SEQUENCE SIZE (1..MAX) OF PKIStatusInfo, 与 RevReqContent 中的次序相同

revCerts [0] SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL, 请求作废的证书的 ID(与 status 次序相同)

crls [1] SEQUENCE SIZE (1..MAX) OF CertificateList OPTIONAL 由此产生的 CRLs(可能不止一个)

}

B.6 证书更新协议

流程参照 5.1.2 节。

申请的 PKIBody 为 CertReqMessages，命令字是 kur。通常情况下，可以为每一个更新的密钥提供 SubjectPublicKeyInfo，KeyId 和 Validity 字段。该消息用于请求更新现有证书(尚未作废且未过期)。

回复的 PKIBody 为 CertRepMessage，命令字是 kup。该消息与证书申请的回复相同。

B.7 证书冻结协议

证书冻结应是证书临时撤销，协议同证书撤销；只在撤销原因项注明原因，表示证书并未作废，有可能重新解冻启用。

B.8 证书解冻协议

证书解冻应是证书临时撤销的反动作，协议同证书撤销；只在撤销原因项注明原因，表示证书解冻并重新启用。

B.9 密钥恢复协议

密钥回复功能现在几乎不放到 RA 来进行，密钥管理系统有一套严格的流程提供密钥回复。RA 这里要提供也只是自己回复自己的加密私钥，前提是保证自己的签名私钥是有效的。申请的 PKIBody 为 CertReqMessages，命令字是 krr。

密钥恢复响应的 PKIBody 为 KeyRecRepContent，令字是 krp。对某些状态值(如 waiting)，可选项都不出现。

```
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
}
```

附 录 C
（规范性附录）
协议报文实例

C.1 PKIMessage通用协议实例

从 PKIMessage 的结构定义知道，构建一份报文的几个部件时除 body 有较多变化外，header、protection、extraCerts 的格式相对固定。

PKIMessage 的定义是这样的 PKIMessage ::= SEQUENCE {

```
    header       PKIHeader,
    body         PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL
}
```

下面先介绍通用消息即确认消息与错误消息

确认消息：

```
30 82 04 c5
 //header
 30 81 c1
  02 01 01
  … …
  a4 03
   04 01
    31
  b3 02 //conf    [19] PKIConfirmContent,  0xb3=0xa0+19
  05 00 //空值
a0 81 84
  03 81 81 00  6c 54 93 … … dd ee 41 7f //确认签名
a1 82 03 72
  30 82 03 6e
 //确认者的签名证书
```

错误消息：

```
30 82 04 c5
 //header
 30 81 c1
  02 01 01
  … …
  a4 03
   04 01
    31
 b7 XX //[23] ErrorMsgContent 0xb7=0xa0+23
  30 XX   //ErrorMsgContent ::= SEQUENCE {
```

```
  30 XX   //PKIStatusInfo ::= SEQUENCE {
status              PKIStatus,
statusString        PKIFreeText     OPTIONAL,
failInfo            PKIFailureInfo  OPTIONAL
}
    02 01 01//PKIStatus
    03 02 00 0e   //PKIFailureInfo
a0 81 84
   03 81 81 00  6c 54 93 … … dd ee 41 7f //错误消息的签名
a1 82 03 72
 30 82 03 6e
 //发送错误消息者的签名证书
```

C.2 证书申请、回应协议报文实例

证书申请消息是最常用的消息，下面的例子包含结构中的所有部件，包括可选项。

```
30 82 07 61  //sequnce
30 60        //1.PKI Header
  02 01 01    //pvno CMP 的当前版本号，取固定值 1
   a4 1b
   30 19
    31 17 30 15 06 03 55 04 03 1e 0e 00 52 00 41 6c e8 51 8c 67 0d 52 a1 56 68 (RA 注册服务器)
    //sender - PKIMessage 发送者的名字。该名字与 senderKID(提供的话)一起应当可以用于验证对该消息的保护。
    //如果发送方实体不知道任何有关 sender 的信息(如，在初始化请求消息中，
    //EE 可能不知道自己的 DN、e-mail name, IP address 等)，那么"sender"字段值必须包含"NULL"；
    //即编码为长度为 0 的 SEQUENCE OF RDN。在这种情况下，senderKID 必须包含一个标识符
    //(即 reference number)通知接收者用于验证消息的相关的共享秘密信息。
  a4 15
   30 13
   31 11 30 0f 06 03 55 04 03 1e 08 00 74 00 65 00 73 00 74(test)
    //recipient - PKIMessage 接收者的名字。
  a0 11
   18 0f 32 30 30 30 30 35 30 36 31 37 30 32 31 35 5a(gtime 20000506170215)
  a1 0f
   30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00
   //protectionAlg - 指定用于保护消息的算法。如果未提供保护比特(注意 KIProtection 可选)，
   //则该字段必须省略；若提供保护比特，则必须提供该字段。
  a4 03
   04 01
   31   //transactionID - 该字段允许响应消息的接收者将响应与先前发送的请求相互关联起来。
        //例如，当 RA 在给定时刻有多个正在处理的请求的情况下。
  //证书申请部
```

```
 a2 82 02 fe
  30 82 02 fa
   30 82 01 ca  //CertReqMsg ::= SEQUENCE {          请求 1
    30 82 01 c6 //CertRequest ::= SEQUENCE {
     02 01 00 //(第 1 份申请的证书)   //certReqId
     30 82 01 bf    //certTemplate
      a4 22
       a0 0f 17 0d 30 30 30 36 33 30 31 36 30 30 30 30 5a
       a1 0f 17 0d 30 32 30 36 33 30 31 36 30 30 30 30 5a
      a5 13
       30 11
        31 0f 30 0d 06 03 55 04 03 1e 06 90 ed 00 b7 50 b2
      //publickey
      a6 81 9f
       30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 00(algrthim)
       03 81 8d 00 30 81 89 02 81 81 00 b1 … … 03 01 00 01

      a9 81 e1 扩展项
       30 0b 06 03 55 1d 0f 04 04 03 02 07 80
       … …
       30 82 01 28  //CertReqMsg ::= SEQUENCE {          请求 2
    30 82 01 24  //CertRequest ::= SEQUENCE {
     02 01 01  //(第 2 份申请的证书)
     30 82 01 1d //certTemplate
      a4 22
       a0 0f 17 0d 30 30 30 36 33 30 31 36 30 30 30 30 5a
       a1 0f 17 0d 30 32 30 36 33 30 31 36 30 30 30 30 5a
      a5 13
       30 11 31 0f 30 0d 06 03 55 04 03 1e 06 90 ed 00 b7 50 b2
      //  无 A6 publickey
      a9 81 e1
       30 0b 06 03 55 1d 0f 04 04 03 02 04 10
        … …
  //证书申请部结束
 a0 81 84     //3.PKI Protection   (OPTIONAL)
  03 81 81 00
   c0 … … 5b  //签名块结束
  a1 82 03 72    //4.PKI extraCerts
  30 82 03 6e
//RA 的签名证书
30 82 03 6a 30 82 02 d3 … …//RA 的签名证书结束
A 回应报文，包含证书与用户加密私钥的数字信封：
30 82 10 09
 30 21
```

```
02 01 01
a4 02
 30 00
a4 02
 30 00
a1 0f
 30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00
a4 03
 04 01 31
 a3 82 0b 84
30 82 0b 80  //CertRepMessage ::= SEQUENCE {
      caPubs      [1] SEQUENCE SIZE (1..MAX) OF Certificate OPTIONAL,
      response        SEQUENCE OF CertResponse   ××××
      }
 30 82 0b 7c  //SEQUENCE OF CertResponse
  30 82 03 f8  //CertResponse ::= SEQUENCE {
   02 01 00  //certReqId          INTEGER,
  // 将该响应与相应的请求进行匹配(请求中未指定 certReqId 值时，取值为-1)
   30 03  //PKIStatusInfo ::= SEQUENCE {
     02 01 00  //PKIStatus ::= INTEGER {
   30 82 03 ec  //CertifiedKeyPair ::= SEQUENCE {
  。。。。。。。。
    a0 82 03 e8  //CertOrEncCert ::= CHOICE {
                            certificate     [0]     Certificate,
                            encryptedCert   [1] EncryptedValue
                          }
    //第一份证书
    30 82 03 e4
     30 82 03 4d
      a0 03 02 01 02 02 03 01 cf 39
      ... ...
    //第一份证书结束
   30 82 07 7c  //CertResponse ::= SEQUENCE {
      02 01 01  //certReqId           INTEGER,
  // 将该响应与相应的请求进行匹配(请求中未指定 certReqId 值时，取值为-1)
      30 03   //PKIStatusInfo ::= SEQUENCE {
       02 01 00  //PKIStatus ::= INTEGER {
      30 82 07 70  //CertifiedKeyPair ::= SEQUENCE {
              certOrEncCert   CertOrEncCert,
              privateKey      [0] EncryptedValue      OPTIONAL,
              publicationInfo [1] PKIPublicationInfo  OPTIONAL
              }
      a0 82 03 e8  //CertOrEncCert ::= CHOICE {
                            certificate     [0] Certificate,
```

```
                                encryptedCert   [1] EncryptedValue
                            }
            //第二份证书
            30 82 03 e4
             30 82 03 4d
              a0 03 02 01 02 02 03 01 cf 3a
             ... ...
            //第二份证书结束
        a0 82 03 80  //privateKey  [0] EncryptedValue      OPTIONAL,
           //第二份证书数字信封 EncryptedValue 类型的数据结构。
              EncryptedValue ::= SEQUENCE {
              intendedAlg   [0] AlgorithmIdentifier  OPTIONAL,
              -- the intended algorithm for which the value will be used
              symmAlg       [1] AlgorithmIdentifier  OPTIONAL,
              -- 用于加密值的对称算法
              encSymmKey    [2] BIT STRING           OPTIONAL,
              -- 加密后的用于加密值的对称密钥
              keyAlg        [3] AlgorithmIdentifier  OPTIONAL,
              -- 用于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(仅对发送方有意义,且仅当发送方将来需要检查
EncryptedValue 才可能使用)
              encValue       BIT STRING
              }
             30 82 03 7c  //EncryptedValue ::= SEQUENCE {
             03 82 03 78 00  //encValue       BIT STRING
               //第二份证书数字信封体
                 75 … … 72
    a0 81 84
    03 81 81 00  //CA 的签名
    //ca 证书
    a1 82 03 d3
     30 82 03 cf
     //ca 证书 开始
      30 82 03 cb
       30 82 03 34
        a0 03 02 01 02 02 01 00
    证书撤销协议报文
    证书撤销的 PKIMessage 报文的 body 报文示例。
    ab 82 03 02     //body  rr      [11] RevReqContent,
     30 82 02 fe    //RevReqContent ::= SEQUENCE OF RevDetails
      30 82 01 cc   //RevDetails ::= SEQUENCE {
       30 82 01 c1  //CertTemplate
        a4 22
```

a0 0f 17 0d 30 30 30 36 33 30 31 36 30 30 30 30 5a
a1 0f 17 0d 30 32 30 36 33 30 31 36 30 30 30 30 5a
a5 13
30 11 31 0f 30 0d 06 03 55 04 03 1e 06 90 ed 00 b7 50 b2
a6 81 9f
30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 00
03 81 8d 00
30 81 89 02 81 81 00 b1 … … 02 03 01 00 01
a9 81 e3
30 0b 06 03 55 1d 0f 04 04 03 02 07 80
//revocationReason OPTIONAL
//badSinceDate OPTIONAL
//crlEntryDetailscrlEntryDetails OPTIONAL
//一份 RevDetails 结束
//第二份 RevDetails 开始
30 82 01 26
30 82 01 1f CertTemplate
a4 22
a0 0f 17 0d 30 30 30 36 33 30 31 36 30 30 30 30 5a
a1 0f 17 0d 30 32 30 36 33 30 31 36 30 30 30 30 5a
a5 13
30 11 31 0f 30 0d 06 03 55 04 03 1e 06 90 ed 00 b7 50 b2
a9 81 e3 30 0b 06 03 55 1d 0f 04 04 03 02 04 10
30 15 … …
//第二份 RevDetails 结束，若有多份证书撤销可以一直罗列下去。

证书撤销的响应 PKIMessage 报文的 body 报文示例。

ac 82 XX XX //body rp [12] RevRepContent,
30 82 XX XX //RevRepContent ::= SEQUENCE {
30 82 XX XX //status SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
-- 与 RevReqContent 中的次序相同
30 03 PKIStatusInfo ::= SEQUENCE {
02 01 00 //PKIStatus ::= INTEGER { 0 表示请求被正确执行
30 82 03 68 XX XX//证书 2

a) 证书更新协议报文

证书更新的 PKIMessage 报文参照证书申请与回应的报文，差别只在 body 的第一字节。

b) 证书冻结协议报文

证书冻结的报文基本同证书撤销，在撤销原因处即 revocationReason 添加原因项，可以参照 CRL 的原因 ，如 revocationReason 是 Bit 值 6 为证书冻结的请求，与 CRL 的区别是 TAG 值不同，CRL 的 TAG 值是 ENUMERATED 型，而这里的是 Bit 型。具体是：03 02 00 04。

证书冻结的回应报文同证书的撤销回应。即 PKIStatus 为 0 表示请求被正确执行。

c) 证书解冻协议报文

证书解冻的报文基本同证书撤销，在撤销原因处即 revocationReason 添加原因项，可以参照 CRL 的原因 ，如 revocationReason 是 Bit 值 8 为证书解冻的请求，与 CRL 的区别是 TAG 值不同，CRL 的 TAG 值是 ENUMERATED 型，而这里的是 Bit 型。具体是：03 02 00 01。

证书解冻的回应报文同证书的撤销回应。即 PKIStatus 为 0 表示请求被正确执行。

C.3 证书查询下载协议报文实例

```
30 4a
 02 01 02 //message id
 63 45
  04 15 //base
   6f 75 3d 63 65 72 74 2c 64 63 3d 69 73 63 2c 64 63 3d 63 6f 6d
 //ou =cert,dc =isc, dc =com
  0a 01 02 //scope whole subtree
  0a 01 00 //dereference alisses
  02 01 00 //size limit
  02 01 00 //time limit
  01 01 00 //types only
  a3 1b //filter type = equality match(3)
   04 09
    62 65 67 69 6e 74 69 6d 65
  //begintime
   04 0e
    32 30 30 36 30 36 31 33 30 30 30 30 30 30
  //20060613000000
  30 00
查询返回的结果
30 82 05 e2
 02 01 02 //message id
 64 82 05 db
  04 27
   73 65 72 69 61 6c 6e 75 6d 62 65 72 3d 63 30 64 31 2c 6f 75 3d 63 65 72 74 2c 64 63 3d
69 73 63 2c 64 63 3d 63 6f 6d
  //serialnumber= c0d1,ou=cert,dc=isc,dc=com
   30 82 05 ae
    30 1e
     04 0b
      6f 62 6a 65 63 74 43 6c 61 73 73
    //objectClass
      31 0f
       04 03
        74 6f 70
      //t  o  p
       04 08
        63 65 72 74 69 6e 66 6f
      //certinfo
    30 16
```

```
 04 0c
  73 65 72 69 61 6c 4e 75 6d 62 65 72
//serialNumber
 31 06
  04 04
   63 30 64 31
 //c0d1
30 0e
 04 02
  63 6e
//cn
 31 08
  04 06
   74 65 73 74 31 34
 //test14
30 26
 04 08
  75 6e 69 74 6e 61 6d 65
//unitname
 31 1a
  04 18
   e9 83 91 e5 b7 9e e5 b8 82 e5 9b bd e5 ae b6 e7 a8 8e e5 8a a1 e5 b1 80
   //“安徽国税”的UTF编码
30 1d
 04 09
  62 65 67 69 6e 74 69 6d 65
//begintime
 31 10
  04 0e
   32 30 30 36 30 36 31 33 30 30 30 30 30 30
 //20060613000000
30 1b
 04 07
  65 6e 64 74 69 6d 65
//endtime
 31 10
  04 0e
   32 30 30 38 30 36 31 32 30 30 30 30 30 30
 //20080612000000
30 24
 04 10
  75 6e 69 71 75 65 49 64 65 6e 74 69 66 69 65 72
//uniqueIdentifier
 31 10
```

```
    04 0e
     30 30 30 30 30 30 30 30 30 30 31 31 30 36
    //00000000001106
   30 0a
    04 04
     6d 61 69 6c
   //mail
    31 02
     04 00
   30 82 04 cc
    04 0f
     75 73 65 72 43 65 72 74 69 66 69 63 61 74 65
   //usercertificate
    31 82 04 b7
     04 82 04 b3
      //usercertificate data
      30 82 04 af 30 82 04 18 a0 03 02 01 02 02 03(查询返回的证书)
```

C.4 OCSP证书状态查询协议报文实例

```
30 75       //  OCSPRequest::=SEQUENCE {
[1]30 73    //  TBSRequest::=SEQUENCE {
[1.0]   //这里好像缺少版本号,因为是V1版，故缺省，V2 V3应仿X.509
[1.1]// 请求者   可选项
[1.x]30 3e          //requestList  SEQUENCE OF Request,
[1.x.1] 30 3c       //Request::=SEQUENCE {
         [1.x.1.1]
         30 3a      //CertID ::= SEQUENCE {
           30 09    //AlgorithmIdentifier
             06 05 2b 0e 03 02 1a 05 00 //oid__sha1
           04 14 // Hash of Issuer's DN
             7c cf 03 d4 78 74 c7 f5 8c c6 86 ba c0 53 4c a2 dd 8c ac 75 //issuerNameHash
           04 14    //Hash of Issuers public key
             c5 af 09 40 f8 94 7c c4 47 82 04 d7 5c 6c 08 b9 27 71 e6 b6 //issuerKeyHash
           02 01 22 //CertificateSerialNumber
         [1.x.1.2]22 xx xx //本版缺省 singleRequestExtensions
    ... ...
[1.x.n]

[1.2]22 31 //   requestExtensions [2] EXPLICIT Extensions OPTIONAL
       30 2f
         30 11
           06 09 2b 06 01 05 05 07 30 01 02 //(oid__id_pkix_ocsp_nonce)(为防重放攻击)
           04 04 35 37 64 37// （接受响应后要校对这里的随机数）
```

```
            30 1a
              06 09 2b 06 01 05 05 07 30 01 04 //oid__id_pkix_ocsp_response
              04 0d
                30 0b
                  06 09 2b 06 01 05 05 07 30 01 01 //oid__id_pkix_ocsp_basic
                30 xx ... ...
                30 xx ... ...
                ... ...
    [2]20 82 xx xx   //   针对[1]的签名
    [2.1]30 82 xx xx
     [2.1.1]30 xx //AlgorithmIdentifier
     [2.1.2]03 81 81 00 ......
     [2.1.3]20 82 xx xx
       30 82 xx xx //查询者签名证书
   OCSP 服务器返回：
   30 82 08 8a
     0a 01 00 //responseStatus 是 ENUMERATED 变量
     20 82 08 83 //responseBytes [0]EXPLICIT ResponseBytes OPTIONAL
       30 82 08 7f // ResponseBytes ::=  SEQUENCE {
         06 09 2b 06 01 05 05 07 30 01 01//responseType   OBJECT IDENTIFIER
         04 82 08 70 //response   OCTET STRING
           30 82 08 6c // BasicOCSPResponse ::= SEQUENCE {
            30 81 93 //tbsResponseData  ResponseData;ResponseData ::= SEQUENCE {
               [1.1]ver
               [1.2]responderID
                    22 16 //ResponderID ::= CHOICE byName[1],byKey[2]
                    04 14 87 ee 0e 84 dd 2f a4 2b d9 45 74 46 15 9e 13 9a 16 e6 0b 89
               [1.3]//producedAt   GeneralizedTime
                    18 0f 32 30 30 32 30 34 32 36 30 39 31 33 32 32 5a //GeneralizedTime
               [1.4]//SEQUENCE OF SingleResponse,
               30 51// SEQUENCE OF SingleResponse
               [1.4.1]//一份证书状态回复
                 30 4f //SingleResponse ::= SEQUENCE {
               [1.4.1.1]//CertID
                   30 3a // CertID::=SEQUENCE {
                     30 09 06 05 2b 0e 03 02 1a 05 00 //hashAlgorithm
                     04 14 7c cf 03 d4 78 74 c7 f5 8c c6 86 ba c0 53 4c a2 dd 8c ac 75
//issuerNameHash
                     04 14 c5 af 09 40 f8 94 7c c4 47 82 04 d7 5c 6c 08 b9 27 71 e6 b6
//issuerKeyHash
                     02 01 22 serialNumber
                   [1.4.1.2]//statue
                            82 00 //CertStatus ::= CHOICE =unknown[2]IMPLICIT UnknownInfo 单
份证书状态
```

[1.4.1.3]//thisUpdate
18 0f 32 30 30 32 30 34 32 36 30 39 31 33 32 32 5a
//thisUpdate:GeneralizedTime
[1.4.1.4]//nextUpdate 本版缺省
[1.5.1.5]//singleExtensions 本版缺省
[1.5]responseExtensions
21 15 //responseExtensions [1] EXPLICIT Extensions OPTIONAL
30 13
30 11
06 09 2b 06 01 05 05 07 30 01 02
//oid__id_pkix_ocsp_nonce 为防重放攻击
04 04 35 63 39 62 //随机数
[2]30 0d 06 09 2a 86 48 86 f7 0d 01 01 05 05 00
//BasicOCSPResponse:AlgorithmIdentifier
[3]03 81 81 00 XX… …XX //signature
[4]20 82 07 3f //certs [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
追加本服务器的证书以及根证书构成的证书链
30 82 07 3b
30 82 03 cb XX XX//证书 1

C.5 密钥恢复协议报文

密钥恢复的 PKIMessage 请求报文参照证书申请与回应的报文，差别只在 body 的第一字节。
回应报文有专门的结构体：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
}
aa 82 XX XX //krp [10] KeyRecRepContent
30 82 XX XX //KeyRecRepContent ::= SEQUENCE {
30 03 PKIStatusInfo ::= SEQUENCE {
02 01 00 //PKIStatus ::= INTEGER { 0 表示请求被正确执行
a2 82 XX XX //keyPairHist [2] SEQUENCE
30 82 XX XX
30 82 XX XX //CertifiedKeyPair
CertifiedKeyPair 结构同证书申请。

附　录　D
（规范性附录）
非实时发放证书协议流程

非实时发放证书协议流程如图 D.1 所示：

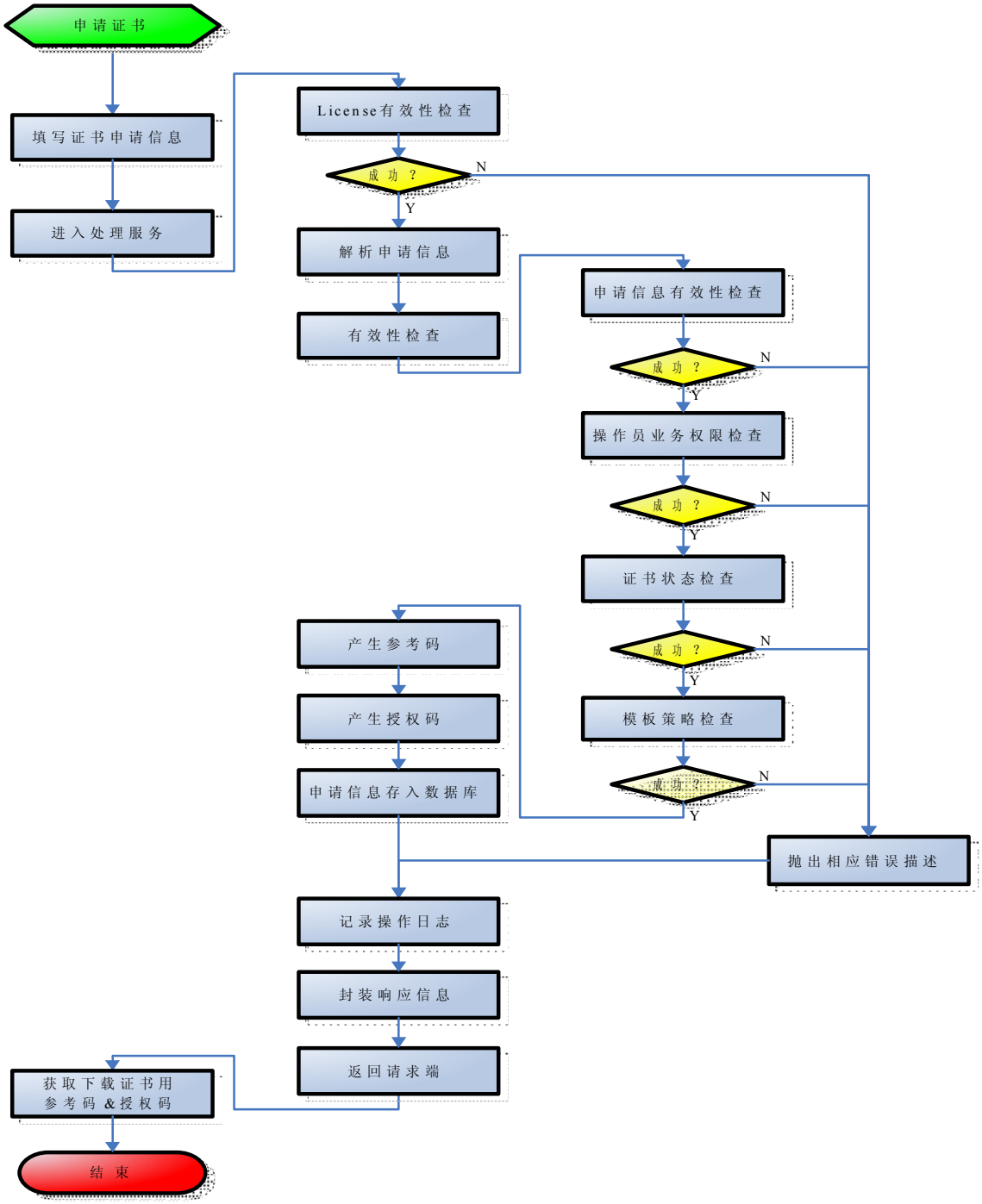

图　D.1 非实时发放证书协议流程