ICS 35.040
L 80

中华人民共和国国家标准

GB/T 29243—2012

信息安全技术　数字证书代理认证路径构造和代理验证规范

Information security technology—Specifications of delegated certification path construction and delegated validation for digital certificate

2012-12-31 发布　　2013-06-01 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

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准由全国信息安全标准化技术委员会(SAC/TC 260)提出并归口。

本标准起草单位:中国科学院数据与通信保护研究教育中心。

本标准主要起草人:夏鲁宁、王琼霄、荆继武、林璟锵、向继。

本标准为首次制定。

引　言

随着《中华人民共和国电子签名法》的推广，我国的电子认证服务业和 PKI 系统的建设应用也进入了新的发展阶段。同时，随着互联网的进一步发展，更多类型的终端接入网络。对于某些类型的终端，如手机、传感器等，由于其计算或通信资源的限制，难以独立完成证书认证路径构造或证书验证，需要 PKI 系统提供代理服务来协助完成上述两种任务。

对于 PKI 依赖方来说，证书认证路径构造和证书验证是必要的过程，但是该过程中所需要的证书查找、撤销信息查找、证书/CRL 验证计算等，需要较大的带宽和计算资源消耗，在计算或通信资源受限的环境下会有不同程度的困难。代理技术是解决上述困难的重要方法，将证书认证路径构造或证书验证委托给代理服务器，能够大大减轻 PKI 客户端的计算负担和通信消耗。

代理认证路径构造和代理验证是两种安全等级不一样的代理服务。对于代理认证路径构造，代理服务器返回验证该证书所需要的完整路径（包括证书链、CRL、OCSP 通信消息等），然后由客户端自己进行验证。这种方式下可以明显减少客户端的通信消耗，且不要求客户端信任服务器；对于代理验证，代理服务器直接返回被验证的证书是否有效。这种方式下客户端的计算负担和通信消耗都明显减少，但客户端应信任代理服务器。为了满足不同交易的安全等级需求，一般要求 PKI 系统同时提供这两种不同的服务。

本标准将定义代理认证路径构造和代理验证两种服务的概念和协议要求，并根据协议要求给出一种标准化的客户端和服务器交互的代理服务协议。

信息安全技术　数字证书代理认证路径构造和代理验证规范

1　范围

本标准规定了数字证书代理认证路径构造和代理验证两种服务的概念和协议要求，以及满足协议要求的代理服务协议。

本标准适用于PKI系统运营机构的代理认证路径构造和代理验证服务的实现和应用。

2　规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 16263.1—2006　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

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

RFC3852　密码消息语法(Cryptographic Message Syntax，CMS)

3　术语和定义

GB/T 16264.8—2005界定的以及以下术语和定义适用于本文件。

3.1

数字证书代理验证　delegated validation for digital certificate

由代理服务器为PKI依赖方实现数字证书验证的过程。

3.2

代理验证　delegated validation

在本标准范围内，与“数字证书代理验证”同义。

3.3

数字证书代理认证路径构造　delegated certification path construction for digital certificate

由代理服务器为PKI依赖方实现数字证书认证路径构造的过程。

3.4

代理认证路径构造　delegated certification path construction

在本标准范围内，与“数字证书代理认证路径构造”同义。

3.5

代理验证策略　delegated validation policy

表达代理验证应如何执行的一系列规则。

3.6

代理认证路径构造策略　delegated certification path construction policy

表达代理认证路径构造应如何执行的一系列规则。

4 缩略语

下列缩略语适用于本文件。

ASN.1　抽象语法标记1(Abstract Syntax Notation One)
CA　证书认证机构(Certification Authority)
CRL　证书撤销列表(Certificate Revocation List)
FTP　文件传输协议(File Transfer Protocol)
HTTP　超文本传输协议(HyperText Transfer Protocol)
LDAP　轻量级目录访问协议(Lightweight Directory Access Protocol)
MAC　消息鉴别码(Message Authentication Code)
OCSP　在线证书状态协议(Online Certificate Status Protocol)
OID　对象标识符(Object Identifier)
PDA　个人数字助理(Personal Digital Assistant)
PKI　公钥基础设施(Public Key Infrastructure)
PKIX　IETF的安全领域的公钥基础设施工作组(Public-Key Infrastructure for X.509)

5 代理服务

5.1 服务基本模式

本标准定义的代理服务采用客户机/服务器模式,通过请求/响应消息对进行,即客户端依照自身需要,构建请求消息,并发往服务器端;服务器端接收客户端的请求消息,提取出请求消息中的具体服务要求并进行相应的处理,最后将处理结果通过响应消息返回给客户端,这就是一次代理服务的全过程。附录A给出了关于代理服务基本原理的细节表述。

在实际环境中,客户端与服务器端的通信过程可能会受到各种类型的攻击,比如伪造消息、篡改消息等,这可能会给代理服务带来严重的影响。因而在客户端与服务器端通信的过程中,应对消息进行保护。通常情况下,可以采取的方式有数字签名和消息鉴别码(MAC),客户端和服务器端根据实际需要选择消息保护类型。

根据服务器端的可信程度的不同,将其划分为可信、不必可信两种情况。每种情况下,服务器端可以提供相应信任程度的代理服务。本标准中规定了两种不同类型的代理服务:代理认证路径构造和代理验证。下面对两种代理服务分别进行说明。

5.2 代理认证路径构造

在代理认证路径构造服务中,服务器端对于客户端不必是可信的。

在代理认证路径构造服务中,客户端在请求消息中提交需要构造认证路径的终端实体证书或对证书的引用,并可选地包含要求采用的代理认证路径构造策略(参见5.4)。服务器端接收到客户端的请求消息后,提取其中的证书和要求采用的代理认证路径构造策略,进行证书认证路径的构造;如果客户端的请求消息中没有包含对代理认证路径构造策略的要求,则服务器端依照自身的默认代理认证路径构造策略进行证书认证路径构造。通过响应消息,服务器端将构造好的证书认证路径或者错误信息返回给客户端。

在代理认证路径构造服务中,客户端对于服务器端返回的证书认证路径构造结果需要进一步验证,才能确定证书的有效性。

5.3 代理验证

在代理验证服务中,服务器端对于客户端是可信的。

在代理验证服务中,客户端需要在请求消息中包含待验证的终端实体证书或对证书的引用,并可选地包含要求采用的代理验证策略(参见 5.4.2)。服务器端接收到客户端的请求消息后,提取出其中的待验证证书和客户端要求采用的代理验证策略,根据这些信息进行证书认证路径的构造以及证书验证,并将验证结果或错误信息通过响应消息返回给客户端。如果客户端请求消息中未包含对代理验证策略的要求,则服务器端采用默认的代理验证策略为客户端提供代理验证服务。

5.4 代理服务策略

5.4.1 代理认证路径构造策略

代理认证路径构造策略是一系列规则,用于表达代理认证路径构造应如何执行。

代理认证路径构造策略是代理验证策略(参见 5.4.2)的一个子集,可以直接引用代理验证策略来表达代理认证路径构造策略,也可以从代理验证策略中提取一些主要元素(如信任锚)而形成代理认证路径构造策略。代理认证路径构造服务的客户端需要自行验证证书,因此它可以在本地应用一些对证书认证路径的验证要求,从而使代理认证路径构造策略更为简单。

代理认证路径构造策略由以下三部分构成:

a) 对证书认证路径的要求;

b) 对撤销的要求;

c) 对终端实体证书的要求。

这些要求的含义与代理验证策略所定义的一致,参见 5.4.2。

5.4.2 代理验证策略

代理验证策略是一系列规则,用于表达代理验证应如何执行。

代理验证策略可以包含若干个信任锚。一个信任锚由一个公钥、一个 CA 名称及一个有效期来定义,并可选地包含其他约束。通常使用 CA 的自签名证书来表示一个信任锚。对每个信任锚可能会定义其他的约束,例如证书策略约束或命名约束。

代理验证策略可以指定针对证书认证路径的另外一些条件,如可为证书认证路径的验证算法提供特定的输入值。代理验证策略还可以指定应用于终端实体证书的其他条件,例如对终端实体命名格式提出要求。

为了成功完成证书验证,一个终端实体与一个信任锚之间应能够找到一个有效的证书认证路径。这意味着路径中的任何证书都没有过期或被撤销,且所有应用于证书认证路径的约束都应得到满足。

代理验证策略由以下三个部分构成:

a) 证书认证路径要求

证书认证路径要求包括一系列信任锚,这些信任锚被用于构造证书认证路径;还包括一些用于证书验证的初始条件,这些条件在 GB/T 16264.8—2005 中定义。

b) 证书撤销要求

证书撤销信息可以通过 CRL、增量 CRL 或 OCSP 响应获得。证书撤销要求针对终端实体证书及 CA 证书指定。对终端实体证书的撤销要求可能与对 CA 证书的撤销要求不同,例如对终端实体证书要求其撤销信息尽可能是最新版本,因此可能需要一个 OCSP 响应;而对 CA 证书,CRLs 可能就已经足够了。

代理验证策略应指明获取撤销信息的方式,可能采用的方式有以下几种:

1） 收集完整的 CRLs；
2） 使用 OCSP 查询撤销信息，并获得 OCSP 响应；
3） 收集增量 CRLs(delta CRLs)及相关的完整的 CRLs；
4） 收集任何可用的撤销信息；
5） 无需收集撤销信息。

c） 终端实体证书要求

代理验证策略可以要求终端实体证书包含特定的扩展，例如要求终端实体证书中包含证书用途扩展。

6 代理服务协议要求

6.1 概述

本章指定了对客户端与代理服务器之间通信协议的设计要求，本标准的第 7 章给出了一种满足本章定义的协议要求的代理服务协议，第三方组织也可以遵照本标准设计其他代理服务协议。如果第三方组织遵照本标准设计代理服务协议，则应满足本章所指定的协议要求。

代理服务器所提供的两种服务中，存在三种请求/响应对，用于客户端与服务器之间的通信，以完成相应的代理服务功能。

a） 代理认证路径构造请求/响应对：客户端向代理服务器发出代理认证路径构造请求，从而可以从服务器获得用于本地证书验证的全部信息，如终端实体证书、CA 证书、完整的 CRLs、增量 CRLs 和 OCSP 信息等。服务器参照客户端要求的或自身默认的代理认证路径构造策略执行代理认证路径构造并通过响应返回已构造的证书认证路径或错误信息。

b） 代理验证请求/响应对：客户端向代理服务器发出代理验证请求，代理服务器参照客户端要求的或自身默认的代理验证策略执行代理验证，并通过响应消息返回验证结果或错误信息。

c） 策略查询请求/响应对：客户端向代理服务器发出请求，服务器返回其所支持的代理认证路径构造策略或代理验证策略。

6.2 代理认证路径构造协议要求

代理认证路径构造协议允许客户端请求服务器收集当前时间下可用的用于本地证书验证的数据元素。根据代理认证路径构造策略，这些数据可以通过各种不同的协议(如 LDAP，HTTP，FTP 或 OCSP)或者通过向多个服务器请求来获得。返回的信息可以用于在当前时间下对一个或多个证书的验证。

客户端应能够明确指定除证书认证路径之外，他们是否还需要终端实体证书和/或 CA 证书的撤销信息。若代理认证路径构造服务器不支持客户端请求中指定的代理认证路径构造策略，服务器应返回错误信息。

代理认证路径构造服务器响应包含 0 个、1 个或多个证书认证路径。每个路径由一组证书构成，从待验证的终端实体证书开始，以信任锚结束。若信任锚是一个自签名证书，则不包含在证书认证路径中。此外，若有需求，每个证书相关的撤销信息应在响应中返回。

默认情况下，代理认证路径构造服务器应为代理认证路径构造请求中的每个终端实体证书返回单个证书认证路径。返回的路径可能需要满足一些客户端提出的要求，如客户端可能要求特殊的证书扩展或名称格式。因此，代理认证路径构造服务器应具有能够为代理认证路径构造请求中的每个终端实体证书获取不止一个证书认证路径的方法。

认证路径构造应按照代理认证路径构造策略进行。代理认证路径构造响应应对以下任一种状态进行说明：

a) 根据代理认证路径构造策略，一个或多个证书认证路径被构造，且具备全部被要求的撤销信息；

b) 根据代理认证路径构造策略，一个或多个证书认证路径被构造，仅具有部分被要求的撤销信息；

c) 根据代理认证路径构造策略，一个或多个证书认证路径被构造，不具备任何被要求的撤销信息；

d) 根据代理认证路径构造策略，无法构造证书认证路径；

e) 路径构造失败，有错误产生。

若无错误产生，返回的响应消息由一个或多个证书认证路径组成。若有需求，证书认证路径中的每个证书相关的撤销信息也包含在响应消息中。

为使客户端能够确认响应中的全部元素都是由期望的代理认证路径构造服务器完成的，可能需要一个可鉴别的响应，例如服务器可能对响应签名或通过底层安全协议实现鉴别。这种鉴别是可选的。

6.3 代理验证协议要求

6.3.1 基本协议

代理验证协议中，根据代理验证策略，服务器可以为客户端验证一个或多个证书的有效性。若代理服务器不支持客户端指定的代理验证策略，服务器应返回一个错误信息。若客户端未指定代理验证策略，服务器响应消息中应说明处理请求所采用的代理验证策略。有些代理验证策略的部分参数是可设置的，协议应允许客户端在请求中包含对这些策略相关参数的设置。

客户端请求可以指定任意的验证时间，服务器应对客户端请求的验证时间获取相应的撤销状态信息。代理验证服务器可能使用各种资源获得撤销状态信息，如 OCSP、CRLs、增量 CRLs 以及来自其他代理服务器的响应消息。在请求所指定的验证时间，若撤销状态信息不可用，服务器应在响应消息中说明证书无效，服务器也可以在响应中包含其他信息说明具体原因。

客户端请求中应包含待验证的证书或证书的引用。代理验证服务器处理客户端请求时应具有待验证的证书，若客户端请求中只提供了对待验证证书的引用，服务器应获取待验证证书并确认它是客户端请求中证书引用所对应的证书。代理验证服务器应在响应消息中包含待验证证书或对它的引用。

代理验证响应消息应说明以下状态信息的任一种情况：

a) 根据代理验证策略，证书有效；

b) 根据代理验证策略，证书无效；

c) 根据代理验证策略，证书的有效性不可知；

d) 由于某些错误，证书的有效性无法确定。

若证书无效，服务器响应中应说明具体原因：

a) 代理验证服务器无法确定证书的有效性，因为无法构造证书认证路径。

b) 代理验证服务器成功地构造了一个证书认证路径，但是根据 GB/T 16264.8—2005 中的验证算法，此路径无效。

c) 证书在当前验证时间无效。若稍后有另一个请求，证书可能被判定为有效的。在证书的一个有效期之前或是证书被挂起的时候，会产生这种情况。

协议应能够防止重放攻击，重放攻击的防范应不能依赖于同步时钟。

代理验证服务器应能够将客户端在请求中提供的文本域复制到服务器响应中。

代理验证响应应与对应的请求绑定，这样客户端能够确认请求中的全部参数在服务器构建响应时都被充分考虑。应允许客户端在请求消息中要求服务器响应包含其他信息，以便不信任代理验证服务

器的依赖方能够确信证书有效性验证被正确地执行。若客户端请求附加此类信息，当证书被验证有效时，响应消息中应包含这些信息；证书验证无效或无法确定证书状态时，响应消息中可以不包含这些信息。在某些情况下，代理验证响应会被转发给其他依赖方，此时为了确保响应与请求的绑定，响应应是自包含的，即响应中包含了对应的请求消息。可以通过将请求中的重要内容复制到响应中实现。

为使客户端能确信证书验证是由期望的代理验证服务器完成，代理验证响应应是可鉴别的，但错误信息（如请求格式错误、或未知的验证策略等）除外。

为使客户端能够向信任相同代理验证服务器的第三方证明证书的验证是正确执行的，代理验证响应应被数字签名，错误信息除外。代理验证服务器的证书应证明代理验证服务器的合法性。

代理验证服务器可以要求客户端鉴别。因此，代理验证请求也应可鉴别。

6.3.2 中继、重定向、组播

在某些网络环境中，尤其是有防火墙的网络中，代理验证服务器可能无法获得所需的全部信息。代理验证服务器可能被配置为可以使用一个或多个其他代理验证服务器提供的信息来满足服务需求。在这种情况下，客户端不知道被请求的服务器使用了其他代理验证服务器的服务，被客户端请求的服务器作为客户端向其他服务器请求服务。与原始客户端不同，代理验证服务器具有适当的计算和存储资源，能够使用中继、重定向或组播机制。

要满足上述需求，协议需要包含可选域和/或扩展以支持中继、重定向或组播机制的使用。代理验证客户端并不需要实现对这些机制的支持。若协议支持这些特性，协议应包含条款以使不支持这些特性的代理验证服务器及客户端能够使用基本的服务。

a) 若代理验证服务器支持中继，代理服务器应提供检测环路的机制。

b) 若协议为代理验证服务器提供将一个请求转发给其他代理验证服务器的能力，协议同时应支持转发信息的机制。

c) 请求和响应中的可选参数也可以支持中继、重定向或组播机制。若代理验证客户端忽略这些可选参数，协议应能为客户端正常提供代理验证服务。若代理验证服务器忽略这些可选参数，也应能够正常提供代理验证服务。

6.4 策略查询协议要求

使用策略查询请求/响应对，代理认证路径构造或代理验证客户端应能够获得对代理认证路径构造策略或代理验证策略的引用，包括默认代理认证路径构造策略或代理验证策略。策略查询响应可以包含对预定义的策略或是预知的策略的引用。

7 代理服务协议

7.1 基本请求/响应消息

7.1.1 概述

代理服务协议遵照第6章的要求设计。客户端需要向服务器端发送请求消息，服务器端接收到客户端的请求消息后，根据客户端请求服务类型的不同，做相应的处理，并将处理结果通过响应消息返回给客户端。协议以统一的方式处理代理验证和代理路径构造，因此这两种服务均采用本节所描述的基本请求/响应消息格式。

基本请求/响应消息应采用GB/T 16263.1—2006定义的DER编码方式进行编码。

7.1.2 客户端请求消息

7.1.2.1 客户端请求消息格式

客户端请求消息封装在 CVRequest 结构中。在发送时，CVRequest 封装在 ContentInfo 中，其 ASN.1 描述如下。

```
ContentInfo{
    contentType   id-ct-scvp-certValRequest,--(1.2.840.113549.1.9.16.1.10)
    content       CVRequest}
```

如果客户端不对请求消息予以保护，则 ContentInfo 的 content 字段就是 CVRequest 结构。

如果客户端对请求予以保护，则 content 字段是 SignedData 结构或 AuthenticatedData 结构，其中封装了 CVRequest。这两类结构的 ASN.1 描述参见 RFC3852。在这种情况下，SignedData 或AuthenticatedData 的 EncapsulatedContentInfo 字段的内容如下：

——eContentType 值为 id-ct-scvp-certValRequest；

——eContent 值为 DER 编码后的 CVRequest。

CVRequest 的 ASN.1 描述如下，其中所有标签均为隐式标签。

```
CVRequest::=SEQUENCE{
  cvRequestVersion          INTEGER DEFAULT 1,
  query                     Query,
  requestorRef        [0]   GeneralNames OPTIONAL,
  requestNonce        [1]   OCTET STRING OPTIONAL,
  requestorName       [2]   GeneralName OPTIONAL,
  responderName       [3]   GeneralName OPTIONAL,
  requestExtensions   [4]   Extensions OPTIONAL,
  signatureAlg        [5]   AlgorithmIdentifier OPTIONAL,
  hashAlg             [6]   OBJECT IDENTIFIER OPTIONAL,
  requestorText       [7]   UTF8String (SIZE (1..256)) OPTIONAL}
```

其中各个字段的含义由 7.1.2.2～7.1.2.11 描述。

7.1.2.2 cvRequestVersion

表示请求的版本号。相应的服务器端响应消息中的版本号应与请求消息中的版本号一致。在本标准的本版本中 cvRequestVersion 字段的值应为 1，以后的更新版本可能会指定其他值。

7.1.2.3 query

表示客户端请求中的查询信息，其类型 Query 的 ASN.1 描述如下，其中所有标签均为隐式标签。

```
Query::=SEQUENCE{
  queriedCerts              CertReferences,
  checks                    CertChecks,
  --注意:标签[0]不使用--
  wantBack            [1]   WantBack OPTIONAL,
  validationPolicy          ValidationPolicy,
  responseFlags             ResponseFlags OPTIONAL,
  serverContextInfo   [2]   OCTET STRING OPTIONAL,
```

validationTime [3] GeneralizedTime OPTIONAL,
intermediateCerts [4] CertBundle OPTIONAL,
revInfos [5] RevocationInfos OPTIONAL,
producedAt [6] GeneralizedTime OPTIONAL,
queryExtensions [7] Extensions OPTIONAL}

Query 类型中各个字段的含义由以下 a)～k)描述。

a) queriedCerts

表示客户端请求进行路径构造或者验证的证书。queriedCerts 字段可以包含多个证书,每个代表一个请求主体。如果包含多个证书,那么所有证书的查询请求都是相同的。其类型 CertReferences 的 ASN.1 描述如下,其中所有标签均为隐式标签。

CertReferences::=CHOICE {
pkcRefs [0] SEQUENCE SIZE (1..MAX) OF PKCReference,
acRefs [1] SEQUENCE SIZE (1..MAX) OF ACReference}

1) pkcRefs

pkcRefs 字段表示引用的公钥证书,其类型 PKCReference 的 ASN.1 描述如下,其中所有标签均为隐式标签。

PKCReference::=CHOICE{
cert [0] Certificate,
pkcRef [1] SCVPCertID}

其中,cert 字段表示公钥证书,pkcRef 字段表示对公钥证书的引用标识。这表明可以将证书实体直接放在 queriedCerts 中,也可以将对这个证书的引用放在其中。如果 queriedCerts 字段包含多个公钥证书,可以一些使用证书实体,另一些使用证书引用。

2) acRefs

acRefs 字段表示引用到的属性证书,其类型 ACReference 的 ASN.1 描述如下,其中所有标签均为隐式标签。

ACReference::=CHOICE{
attrCert [2] AttributeCertificate,
acRef [3] SCVPCertID}

其中,attrCert 字段表示属性证书,acRef 字段表示属性证书的引用标识。这表明可以将证书实体直接放在 queriedCerts 中,也可以将对这个证书的引用放在其中。如果 queriedCerts 字段包含多个属性证书,可以一些使用证书实体,另一些使用证书引用。

3) 证书引用标识

对于公钥证书或属性证书的引用标识,其类型 SCVPCertID 的 ASN.1 描述如下。

SCVPCertID::=SEQUENCE{
certHash OCTET STRING,
issuerSerial SCVPIssuerSerial,
hashAlgorithm AlgorithmIdentifier DEFAULT {algorithm sha-1}}

其中,certHash 字段表示对证书的散列值;issuerSerial 字段表示公钥证书或者属性证书的签发者和序列号;hashAlgorithm 字段表示采用的散列算法,默认为 algorithm sha-1。hashAlgorithm 字段指定的散列算法应是服务器策略配置响应消息(参见 7.2.3)的 hashAlgorithms 字段所指定的多个散列算法之一。

SCVPIssuerSerial 类型的 ASN.1 描述如下:

SCVPIssuerSerial::=SEQUENCE {

issuer GeneralNames,
serialNumber CertificateSerialNumber}

其中,issuer 字段表示证书的签发者,serialNumber 字段表示证书的序列号。

b) checks

表示客户端请求服务器端对 queriedCerts 字段中的证书做哪些方面的处理,其类型 CertChecks 的 ASN.1 描述如下:

CertChecks::=SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER

对于公钥证书,本标准中定义以下处理:

——id-stc-build-pkc-path:对一个信任锚构造期望的证书路径;

——id-stc-build-valid-pkc-path:对一个信任锚构造期望的证书路径,并验证有效,不要求检查撤销状态;

——id-stc-build-status-checked-pkc-path:对一个信任锚构造期望的证书路径并验证有效,要求检查撤销状态。

提供代理路径构造服务的代理服务器应支持 id-stc-build-pkc-path;提供代理验证服务的代理服务器应支持以上三种处理。

对于属性证书,本标准中定义以下处理:

——id-stc-build-aa-path:对一个属性证书(AC)颁发者的信任锚构造一个期望的证书路径;

——id-stc-build-valid-aa-path:对一个 AC 颁发者的信任锚构造一个期望的证书路径并验证有效,不要求检查撤销状态;

——id-stc-build-status-checked-aa-path:对一个 AC 颁发者的信任锚构造一个期望的证书路径并验证有效,要求检查路径中除 AC 自己的证书外其他证书的撤销状态;

——id-stc-status-check-ac-and-build-status-checked-aa-path:对一个 AC 颁发者的信任锚构造一个期望的证书路径并验证有效,要求检查所有证书的撤销状态,包括 AC 颁发者。

代理服务器是否支持对属性证书的以上处理,是可选的。

所有的检查类型都使用 OID 表示,描述如下:

id-stc OBJECT IDENTIFIER::={iso (1) identified-organization (3) dod (6) internet (1) security (5) mechanisms (5) pkix (7) 17}

id-stc-build-pkc-path OBJECT IDENTIFIER::={id-stc 1}
id-stc-build-valid-pkc-path OBJECT IDENTIFIER::={id-stc 2}
id-stc-build-status-checked-pkc-path OBJECT IDENTIFIER::={id-stc 3}
id-stc-build-aa-path OBJECT IDENTIFIER::={id-stc 4}
id-stc-build-valid-aa-path OBJECT IDENTIFIER::={id-stc 5}
id-stc-build-status-checked-aa-path OBJECT IDENTIFIER::={id-stc 6}
id-stc-status-check-ac-and-build-status-checked-aa-path OBJECT IDENTIFIER::={id-stc 7}

c) wantBack

wantBack 字段可选,表示客户端请求服务器端除返回 checks 中所定义的处理结果外,还要返回什么内容。其类型 WantBack 的 ASN.1 描述如下:

WantBack::=SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER

若存在 wantBack 项,wantBack 应包含一系列 OIDs。每个 OID 表示客户端请求服务器返回关于 queriedCerts 的哪些信息。对于 wantBack 中指定的每种信息,服务器都应找到相应的结果并返回。例如,请求的 checks 字段指定构造证书路径(id-stc-build-pkc-path),wantBack 要求返回服务器构造的所有证书路径(id-swb-pkc-all-cert-paths)。在这种情况下,响应不会包含证书路径的验证状态,这就是典型的代理路径构造服务。

对公钥证书，定义了以下几种 wantBacks：

——id-swb-pkc-cert：请求中的主体证书，即 queriedCerts 字段所指示的证书；

——id-swb-pkc-best-cert-path：最优的证书路径，路径中每个证书都是被验证有效的；

——id-swb-pkc-revocation-info：证书路径中每个证书的撤销状态证明；

——id-swb-pkc-public-key-info：请求中主体证书的公钥；

——id-swb-pkc-all-cert-paths：为请求中主体证书构造的所有证书路径；

——id-swb-pkc-ee-revocation-info：路径中终端实体证书的撤销状态证明；

——id-swb-pkc-CAs-revocation-info：路径中每个 CA 证书的撤销状态证明。

提供代理验证服务的代理服务器应支持 id-swb-pkc-cert 和 id-swb-pkc-public-key-info；提供代理路径构造服务的代理服务器应支持 id-swb-pkc-cert、id-swb-pkc-public-key-info、id-swb-pkc-best-cert-path 和 id-swb-pkc-revocation-info。

对属性证书，定义了以下几个 wantBacks：

——id-swb-ac-cert：请求中的主体属性证书；

——id-swb-aa-cert-path：对 AC 颁发者证书构造的证书路径；

——id-swb-ac-revocation-info：AC 颁发者证书路径中每个证书的撤销状态证明；

——id-swb-aa-revocation-info：属性证书的撤销状态证明。

代理服务器对以上属性证书 wantBacks 的实现是可选的。

另外，还有一个 wantBacks 在公钥证书和属性证书中都可以使用：

——id-swb-relayed-responses：代理服务器为了完成客户端指定的处理而从其他代理服务器获得的响应信息。

代理服务器对这个 wantBack 的支持是可选的。

每一个 wantBack 都使用一个 OID 表示。因此定义了以下 OID：

id-swb OBJECT IDENTIFIER::={iso (1) identified-organization (3) dod (6) internet (1) security (5) mechanisms (5) pkix (7) 18}

id-swb-pkc-best-cert-path OBJECT IDENTIFIER::={id-swb 1}

id-swb-pkc-revocation-info OBJECT IDENTIFIER::={id-swb 2}

id-swb-pkc-public-key-info OBJECT IDENTIFIER::={id-swb 4}

id-swb-aa-cert-path OBJECT IDENTIFIER::={id-swb 5}

id-swb-aa-revocation-info OBJECT IDENTIFIER::={id-swb 6}

id-swb-ac-revocation-info OBJECT IDENTIFIER::={id-swb 7}

id-swb-relayed-responses OBJECT IDENTIFIER::={id-swb 9}

id-swb-pkc-cert OBJECT IDENTIFIER::={id-swb 10}

id-swb-ac-cert OBJECT IDENTIFIER::={id-swb 11}

id-swb-pkc-all-cert-paths OBJECT IDENTIFIER::={id-swb 12}

id-swb-pkc-ee-revocation-info OBJECT IDENTIFIER::={id-swb 13}

id-swb-pkc-CAs-revocation-info OBJECT IDENTIFIER::={id-swb 14}

d) validationPolicy

表示客户端要求服务器端使用的代理验证策略或代理路径构造策略。本协议中对这两类代理服务的策略不做明确区分，而是以统一的格式来定义，后面描述中均以“策略”指代。如果服务器端不支持客户端指定的策略，则应在响应中返回错误信息。策略应为处理客户端请求所需的每个参数定义默认值。对每个参数，策略可以指示允许或禁止客户端指定非默认值。如果客户端希望对全部参数使用默认值，则只需使用该策略的引用即可；如果客户端希望对一个或多个参数使用非默认值，则需在本字段中包含该策略的引用及全部参数。若客户端指定的策略与使用的参数值有冲突，服务器应在响应中返回错误

信息。

类型 ValidationPolicy 的 ASN.1 描述如下，其中所有标签均为隐式标签。

```
ValidationPolicy::=SEQUENCE {
validationPolRef                ValidationPolRef,
validationAlg           [0]     ValidationAlg OPTIONAL,
userPolicySet           [1]     SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER
                                OPTIONAL,
inhibitPolicyMapping    [2]     BOOLEAN OPTIONAL,
requireExplicitPolicy   [3]     BOOLEAN OPTIONAL,
inhibitAnyPolicy        [4]     BOOLEAN OPTIONAL,
trustAnchors            [5]     TrustAnchors OPTIONAL,
keyUsages               [6]     SEQUENCE OF KeyUsage OPTIONAL,
extendedKeyUsages       [7]     SEQUENCE OF KeyPurposeId OPTIONAL,
specifiedKeyUsages      [8]     SEQUENCE OF KeyPurposeId OPTIONAL}
```

ValidationPolicy 类型中各个字段的含义由以下 1)～10)描述。

1) validationPolRef

客户端和服务器端都认可的策略，用 OID 和相应的参数表示。

ValidationPolRef 的 ASN.1 描述如下：

```
ValidationPolRef::=SEQUENCE {
    valPolId                    OBJECT IDENTIFIER,
    valPolParams                ANY DEFINED BY valPolId OPTIONAL}
```

其中，valPolId 字段表示策略的 OID，valPolParams 字段表示相应的参数。若策略允许客户端指定参数，这些参数值用 valPolParams 描述。若策略没有参数，那么 valPolParams 应被省略。

用户可以请求使用服务器默认的策略。默认的策略使用服务器选择的默认值进行操作，并且遵循 GB/T 16264.8—2005 定义的标准证书验证过程。默认值可以带外发布，也可以使用策略请求(参见 7.2.2)机制发布。默认的策略总是使用同一个 OID 表示，即下列定义中的 id-svp-defaultValPolicy：

```
id-svp OBJECT IDENTIFIER::={iso (1) identified-organization (3) dod (6) internet (1) security (5) mechanisms (5) pkix (7) 19}
id-svp-defaultValPolicy OBJECT IDENTIFIER::={id-svp 1}
```

默认策略应使用与默认验证算法相同的基本验证算法，并且不使用参数。代理服务器应支持默认策略。

2) validationAlg

表示客户端要求服务器端在进行证书验证时用的验证算法，这个算法可能需要双方协商确定。其类型 ValidationAlg 的 ASN.1 描述如下：

```
ValidationAlg::=SEQUENCE{
    valAlgId                    OBJECT IDENTIFIER,
    parameters                  ANY DEFINED BY valAlgId OPTIONAL}
```

其中，valAlgId 字段表示算法 OID，parameters 字段表示相应的参数值。

代理服务器应能识别并支持基本验证算法和命名验证算法。

——基本验证算法

客户端可以请求代理服务器使用基本验证算法。对于公钥证书，基本验证算法应实现 GB/T 16264.8—2005 定义的公钥证书验证算法。对于属性证书，基本验证算法应实现 GB/T 16264.8—2005 定义的属性证书验证算法。

如果客户端请求中指定了非基本验证算法的其他算法，则该验证算法应产生与基本验证算法兼容的结果。换言之，基本验证算法的功能是任何其他验证算法的功能子集。

基本验证算法的 OID 定义如下：

id-svp-basicValAlg OBJECT IDENTIFIER::={id-svp 3}

当 valAlgId 的值是 id-svp-basicValAlg 时，parameters 字段应忽略。

对应于基本验证算法，还定义了以下错误信息。如果客户端请求指定了基本验证算法，那么这些错误信息将可能出现在服务器响应的 replyObjects 的 validationErrors 字段[参见 7.1.3.10f)]。如果客户端请求指定了其他验证算法，也可能产生这些错误，因为任何其他算法都应实现基本验证算法的功能。

id-bvae OBJECT IDENTIFIER::=id-svp-basicValAlg
id-bvae-expired OBJECT IDENTIFIER::={id-bvae 1}
id-bvae-not-yet-valid OBJECT IDENTIFIER::={id-bvae 2}
id-bvae-wrongTrustAnchor OBJECT IDENTIFIER::={id-bvae 3}
id-bvae-noValidCertPath OBJECT IDENTIFIER::={id-bvae 4}
id-bvae-revoked OBJECT IDENTIFIER::={id-bvae 5}
id-bvae-invalidKeyPurpose OBJECT IDENTIFIER::={id-bvae 9}
id-bvae-invalidKeyUsage OBJECT IDENTIFIER::={id-bvae 10}
id-bvae-invalidCertPolicy OBJECT IDENTIFIER::={id-bvae 11}

id-bvae-expired 表示验证时间晚于终端实体证书(即客户端请求的 queriedCerts 中的证书)的 notAfter 项所指定的时间。

id-bvae-not-yet-valid 表示验证时间早于终端实体证书的 notBefore 项所指定的时间。

id-bvae-wrongTrustAnchor 表示无法根据客户端指定的信任锚构造证书路径。但可以以服务器默认验证策略中指定的信任锚构造证书路径。

id-bvae-noValidCertPath 表示服务器无法构造满足客户端请求中指定的各种检查和约束的证书路径。

id-bvae-revoked 表示终端实体证书已经被撤销。

id-bvae-invalidKeyPurpose 表示终端实体证书中的扩展的密钥用法(Extended Key Usage)扩展项不能满足策略。

id-bvae-invalidKeyUsage 表示终端实体证书中的密钥用法(Key Usage)扩展项不能满足策略。

id-bvae-invalidCertPolicy 表示无法构造路径，因为路径中某个证书的 valid_policy_tree 是 NULL 且 explicit_policy 是 0。

——命名验证算法

命名验证算法除了具有基本验证算法的功能外，还允许客户端指定一个或多个应在终端证书中出现的主体名称。命名验证算法使用如下 OID 表示。

id-svp-nameValAlg OBJECT IDENTIFIER::={id-svp 2}

当使用命名验证算法，即 valAlgId 的值为 id-svp-nameValAlg 时，parameters 应使用下面定义的 NameValidationAlgParms 语法：

```
NameValidationAlgParms::=SEQUENCE{
    nameCompAlgId    OBJECT IDENTIFIER,
    validationNames  GeneralNames}
```

若在 validationNames 中使用了多个名称，这些名称应类型相同。证书应对每个在 validationNames 中使用的名称包含一个根据 nameCompAlgId 所述的名称匹配规则相符的名称。

本标准定义了三种名称匹配规则：

id-nva-dnCompAlg　　OBJECT IDENTIFIER::={id-svp 4}
id-kp-serverAuth　　OBJECT IDENTIFIER::={id-kp 1}(参见 GB/T 16264.8—2005)
id-kp-emailProtection　　OBJECT IDENTIFIER::={id-kp 4}(参见 GB/T 16264.8—2005)

如果 nameCompAlgId 的值为 id-nva-dnCompAlg,客户端请求提供的 GeneralNames 应是 directoryName(参见 GB/T 16264.8—2005 中 GeneralNames 的定义)。证书应在主体字段或 subjectAltName 的 directoryName 中包含一个匹配的名称。

如果 nameCompAlgId 的值为 id-kp-serverAuth,客户端请求提供的 GeneralNames 应是 dNSName(参见 GB/T 16264.8—2005 中 GeneralNames 的定义)。如果证书包含了 subjectAltName 扩展并且其中有多个 dNSName 类型的名称,则与任何一个名称匹配都可以。如果证书不包含 subjectAltName 扩展或扩展中没有 dNSName 类型的名称,则主体字段中的 Common Name 应被使用。名称可以包含通配符“*”,例如“*.a.com”能够匹配“foo.a.com”,但不能匹配“bar.foo.a.com”。

如果 nameCompAlgId 的值为 id-kp-emailProtection,客户端请求提供的 GeneralNames 应是 rfc822Name(参见 GB/T 16264.8—2005 中 GeneralNames 的定义)。

代理服务器应支持命名验证算法的以上三种名称匹配形式。

对应于命名验证算法,还定义了以下错误信息:

id-nvae　　OBJECT IDENTIFIER::=id-svp-nameValAlg
id-nvae-name-mismatch　　OBJECT IDENTIFIER::={id-nvae 1}
id-nvae-no-name　　OBJECT IDENTIFIER::={id-nvae 2}
id-nvae-unknown-alg　　OBJECT IDENTIFIER::={id-nvae 3}
id-nvae-bad-name　　OBJECT IDENTIFIER::={id-nvae 4}
id-nvae-bad-name-type　　OBJECT IDENTIFIER::={id-nvae 5}
id-nvae-mixed-names　　OBJECT IDENTIFIER::={id-nvae 6}

id-nvae-name-mismatch 表示服务器在证书中找到与客户端提供名称同类型的名称,但没有完全相匹配。例如客户端提供了一个 dNSName“example1.com”,但证书中的是“example.com”。

id-nvae-no-name 表示服务器无法在证书中找到与客户端提供的名称同类型的名称。例如客户端提供了一个 dNSName“example1.com”,但证书中只包含 rfc822Name“user@example.com”。

id-nvae-unknown-alg 表示服务器无法识别客户端提供的 nameCompAlgId。

id-nvae-bad-name 表示客户端提供的是空名称或格式错误的名称。

id-nvae-bad-name-type 表示客户端提供了与 nameCompAlgId 指定的匹配算法不一致的名称,例如客户端指定 nameCompAlgId 为 id-kp-serverAuth,但提供的名称是 rfc822Name“user@example.com”。

id-nvae-mixed-names 表示客户端提供了多种不同类型的名称。

3)　userPolicySet

表示一组证书策略标识符,代理服务器在构造证书路径和验证证书的时候应使用。其可能值等同于 GB/T 16264.8—2005 中定义的 user-initial-policy-set。

代理服务器应能够识别并处理客户端请求中的 userPolicySet。

4)　inhibitPolicyMapping

inhibitPolicyMapping 是证书验证算法的一个输入参数,表示在进行证书路径构造和验证的过程中是否允许进行策略映射。

5)　requireExplicitPolicy

requireExplicitPolicy 是证书验证算法的一个输入参数,表示证书策略扩展中是否至少应有一个有效的策略。若用户希望向服务器至少请求一种证书策略,requireExplicitPolicy 在请求中被置为“True”。

6） inhibitAnyPolicy

inhibitAnyPolicy 是证书验证算法的一个输入参数，表示在评估证书策略时，anyPolicy OID 是被执行还是被忽略。若用户希望服务器忽略 anyPolicy OID，inhibitAnyPolicy 被置为“True”。

7） trustAnchors

trustAnchors 表示代理服务器应使用的信任锚。若请求中存在 trustAnchors，那么服务器一定不可以判定以其他信任锚为终点的证书路径有效。

TrustAnchors 类型有一种或多种信任锚的描述，可以使用证书引用，也可以直接提供证书实体。任何符合 GB/T 16264.8—2005 的 CA 证书都可以作为信任锚。若所使用的信任锚不符合以上要求，服务器应在响应中返回错误信息。

信任锚本身一定不可以出现在代理服务器返回的任何路径中。

TrustAnchors 类型的 ASN.1 描述如下：

TrustAnchors::=SEQUENCE SIZE (1..MAX) OF PKCReference

代理服务器应支持 trustAnchors。

8） keyUsages

表示客户端要求服务器端在进行证书路径构造或证书验证的过程中检查密钥用途。GB/T 16264.8—2005 中定义的 keyUsage 扩展，表示证书所含密钥的用途（如加密、签名、CRL 签名）。例如，如果客户端从数字签名中获得了签名者的证书，则可以通过包含一个数字签名位被置位的 keyUsage 结构来请求服务器验证此证书是否可被用作数字签名。

如果 keyUsages 中包含了多个 keyUsage 项，则证书应满足至少一个指定的 keyUsage。若证书中不存在 keyUsage 扩展，则证书应被认为是能满足全部用途。

代理服务器应能够识别并处理 keyUsages。

9） extendedKeyUsages

表示客户端要求服务器端在进行证书路径构造或证书验证的过程中允许的扩展密钥用途。证书中的扩展密钥用途扩展定义了更多的密钥用途，这些用途可能是替代 keyUsage 扩展中指定的密钥用途或是在其基础上进行了增加。KeyPurposeId 类型实质上是 OBJECT IDENTIFIER，定义如下：

KeyPurposeId::=OBJECT IDENTIFIER（参见 GB/T 16264.8—2005）

如果证书中没有 extendedKeyUsage 扩展，或 extendedKeyUsage 扩展的值为 anyExtendedKeyUsage，则客户端请求中指定的所有 extendedKeyUsages 都被认为是匹配的；如果证书中含有 extendedKeyUsages 且不为 anyExtendedKeyUsage，则客户端在 extendedKeyUsages 中指定的所有扩展密钥用途项应全部在证书的 extendedKeyUsage 中出现。

如果客户端无须服务器检查扩展密钥用途，则可以为 extendedKeyUsages 指定一个空 SEQUENCE。这可以用来覆盖 validationPolRef 的 valPolId 所指定的扩展证书用途。

代理服务器应能够识别并处理 extendedKeyUsages。

10） specifiedKeyUsages

与 extendedKeyUsages 类似，表示客户端要求服务器端在进行证书路径构造和验证的过程中允许的扩展密钥用途。若请求中包含 specifiedKeyUsages 且指定一个或多个扩展密钥用途，则证书应包含 extendedKeyUsage 扩展，且客户端请求的 specifiedKeyUsages 中指定的所有扩展密钥用途应全部在证书的 extendedKeyUsage 中出现。如果证书的 extendedKeyUsage 值为 anyExtendedKeyUsage，则被认为无法满足客户端指定的扩展密钥用途。

代理服务器应能够识别并处理 specifiedKeyUsages。

e） responseFlags

表示客户端要求服务器端返回的响应消息中包含哪些特征。这个字段是可选的，如果所有的标志都为默认值，那么在请求中一定不能包含 responseFlags 字段。类型 ResponseFlags 的 ASN.1 描述如

下，其中所有标签均为隐式标签。

```
ResponseFlags::=SEQUENCE{
    fullRequestInResponse          [0]  BOOLEAN DEFAULT FALSE,
    responseValidationPolByRef     [1]  BOOLEAN DEFAULT TRUE,
    protectResponse                [2]  BOOLEAN DEFAULT TRUE,
    cachedResponse                 [3]  BOOLEAN DEFAULT TRUE}
```

1） fullRequestInResponse

默认情况下，代理服务器的非缓存响应消息会包含客户端请求的散列值，以便客户端鉴别。如果fullRequestInResponse 设置为 TRUE，则服务器响应中包含完整的客户端请求消息。

代理服务器应支持返回完整的客户端请求消息。

2） responseValidationPolByRef

responseValidationPolByRef 决定响应中是仅包含对策略的引用，还是除策略引用外还包含用于处理请求的全部参数的值。响应中包含对策略的引用是应的，若用户希望在响应中还包含策略的参数值，则 responseValidationPolByRef 被置为“FALSE”。

3） protectResponse

表示客户端要求服务器对响应消息予以保护。代理服务器默认会对响应消息用数字签名或 MAC 进行保护，如果这个项设置为 FALSE，则说明客户端指示响应消息无须上述保护。

代理路径构造服务的客户端应支持将 protectResponse 的值设为“FALSE”。而代理验证服务的客户端一般需要服务器对响应消息予以保护，即将 protectResponse 设为“TRUE”或省略。如果通信采用了其他保护方式，例如 TLS，则客户端可以将此项设置为“FALSE”。

4） cachedResponse

说明客户端是否会接受经缓存的响应。cachedResponse 被置为“TRUE”时，客户端接受经缓存的响应。cachedResponse 被置为“FALSE”时，客户端不接受经缓存的响应。为确保接收到的响应是新产生的，客户端应在请求中包含 requestNonce（参见 7.1.2.5）。

代理服务器应支持 cachedResponse。若服务器无法产生新的响应时，应返回错误信息。

f） serverContextInfo

serverContextInfo 包含客户端与代理服务器前一次通信的上下文信息，这些信息来自前一次通信的服务器响应。使用这个字段可以使服务器为同一个证书返回多条路径。例如，代理服务器为客户端请求的证书构造了一条路径，但客户端发现不可接受，于是客户端重新发送了相同的请求并设置 serverContextInfo 为上一次的服务器响应中 serverContextInfo 项的内容（参见 7.1.3.12），于是代理服务器这次会返回一条不同的证书路径（如果有）。如果客户端这样做，那么前后两次请求除以下字段外，都应该是相同的：

——requestNonce

——serverContextInfo

——客户端对请求信息的数字签名或 MAC

代理服务器应支持 serverContextInfo。

g） validationTime

表示客户端要求服务器执行 checks 中指定的处理的时间。如果不提供 validationTime，则代理服务器以服务器上的当前时间为准。如果指定了 validationTime，则这个时间应在当前时间之前。

h） intermediateCerts

表示客户端提供给服务器端在证书路径构造过程中可能用到的中间证书。服务器在路径构造中如果发现有多条路径，则尽量选择包含 intermediateCerts 中证书的路径。其类型 CertBundle 的 ASN.1 描述如下。

CertBundle::=SEQUENCE SIZE (1..MAX) OF Certificate

代理服务器应能够识别并处理 intermediateCerts,但并不假定这些证书一定是有效的。

i) revInfos

表示客户端提供给服务器在证书验证过程中可能用到的撤销信息,其类型 RevocationInfos 的 ASN.1 描述如下,其中所有标签均为隐式标签。

RevocationInfos::=SEQUENCE SIZE (1..MAX) OF RevocationInfo

RevocationInfo::=CHOICE {

crl	[0]	CertificateList,
delta-crl	[1]	CertificateList,
ocsp	[2]	OCSPResponse,
other	[3]	OtherRevInfo}

其中,crl 字段表示证书撤销列表,delta-crl 字段表示增量证书撤销列表,ocsp 字段表示 OCSP 服务器的响应消息,other 字段表示其他类型的撤销信息。

OtherRevInfo 类型的 ASN.1 描述如下:

OtherRevInfo::=SEQUENCE{

riType	OBJECT IDENTIFIER,
riValue	ANY DEFINED BY riType}

其中,riType 字段表示撤销信息的 OID,riValue 字段表示撤销信息的值。

j) producedAt

客户端如果允许服务器使用经缓存的响应(在 cachedResponse 设置),则可以使用 producedAt 来要求缓存响应的时效程度。这是一个时间值,只有形成时间晚于此值的缓存响应才被客户端接受。

支持响应缓存的代理服务器应支持 producedAt;不支持响应缓存的代理服务器可以选择不支持 producedAt。

k) queryExtensions

表示客户端请求消息中对 query 的一些扩展信息。本标准中不定义任何扩展,此项保留为以后使用。其类型 Extensions 的 ASN.1 描述如下:

Extensions::=SEQUENCE SIZE (1..MAX) OF Extension

Extension::=SEQUENCE {

extnID	OBJECT IDENTIFIER,
critical	BOOLEAN DEFAULT FALSE,
extnValue	OCTET STRING }

其中,extnID 是标识此扩展的 OID。critical 字段表示扩展的关键程度,如果是关键扩展则为 TRUE,否则为 FALSE,默认为 FALSE。如果是关键扩展但代理服务器无法识别,则应拒绝此请求;如果是非关键扩展,则服务器无法识别时忽略此扩展。extnValue 字段表示扩展值。

7.1.2.4 requestorRef

requestorRef 是一个代理服务器的列表,当代理服务需要中继(代理服务器需要从其他代理服务器获取信息)时会用到。有关中继的应用已在 6.3.2 中描述。

代理服务器应能够识别 requestorRef。若客户端在请求中包含了 requestorRef,则在非缓存响应中代理服务器应返回相同的值,在缓存响应中则可以不返回。

7.1.2.5 requestNonce

requestNonce 包含客户端产生的一个随机数。如果客户端在请求中包含了 requestNonce,则服务

器应返回一个非缓存的响应。在此非缓存响应中,应包含与请求中相同的 requestNonce。

requestNonce 可以被客户端用来防范重放攻击,代理服务器应能够识别并处理 requestNonce。如果客户端请求包含了 requestNonce 同时设置 cachedResponse 为 FALSE,则客户端要求服务器应返回包含此 requestNonce 的非缓存响应,或返回错误信息。requestNonce 只能使用一次,后续请求中如包含 requestNonce,则需使用另外的随机数。

7.1.2.6 **requestorName**

表示客户端名称。代理服务器应能够识别 requestorName。

7.1.2.7 **responderName**

表示客户端希望对响应消息进行签名的代理服务器的名称。只有在下列两种情况都满足的条件下才允许包含 responsderName:

——客户端请求没有得到保护(客户端没有对请求信息签名或计算 MAC);

——responseFlags 项没有设置,或者设置了但 protectResponse 设为 TRUE,即客户端要求响应消息需要被保护。

7.1.2.8 **requestExtensions**

表示请求消息中的扩展信息,其类型 Extensions 的定义与 7.1.2.3k)的定义相同。如果请求信息中包含了 requestExtensions,则其中的每一项表示对请求的一个扩展。本标准不定义任何扩展,此项保留为未来使用。

7.1.2.9 **signatureAlg**

表示客户端要求服务器端对响应消息进行签名时所用的签名算法。只有在下列两种情况都满足的条件下才允许包含 signatureAlg:

——客户端请求没有得到保护(客户端没有对请求信息签名或计算 MAC);

——responseFlags 项没有设置,或者设置了但 protectResponse 设为 TRUE,即客户端要求响应消息需要被保护。

signatureAlg 可以指定的签名算法在策略配置响应消息的 signatureGeneration(参见 7.2.3.16)列出。代理服务器应能够识别 signatureAlg。

如果 signatureAlg 指定了策略配置响应消息的 signatureGeneration 中包含的一个算法,则服务器应使用这个算法来对响应签名。

如果请求消息中没有 signatureAlg 项,或指定了服务器不支持的签名算法,则服务器应使用 signatureGeneration 中指定的默认签名算法对响应消息签名。

7.1.2.10 **hashAlg**

表示客户端要求服务器在计算请求信息的散列值时使用的散列算法,这个散列值会包含在服务器响应消息的 requestHash 中。hashAlg 指定的散列算法应在策略配置响应消息的 hashAlgorithms 中有定义。如果 query 的 responseFlags 指定 fullRequestInResponse 为 TRUE,则一定不能设置 hashAlg 项,且服务器响应中也不能出现 requestHash。

代理服务器应能够识别 hashAlg。

若请求消息中 hashAlg 指定的算法与策略配置响应消息中的 hashAlgorithms 说明的算法一致,服务器应使用指定的算法进行计算散列值。

若 hashAlg 不存在,或服务器不支持 hashAlg 指定的算法,服务器应使用策略配置响应消息中的

hashAlgorithms 说明的默认散列算法计算散列值。

7.1.2.11 requestorText

表示客户端提供的文本信息。客户端可能要求服务器端的响应中包含这些文本信息。代理服务器应能够识别 requestorText。如果服务器端针对一个请求产生非缓存响应，则在服务器响应信息的 requestorText 中应包含相同的文本信息。

7.1.3 服务器端响应消息

7.1.3.1 服务器端响应消息格式

服务器端接收到客户端的请求消息后，根据客户端的请求进行证书路径构造或证书验证，并将结果通过响应消息返回给客户端。响应消息封装在 CVResponse 结构中，CVResponse 又封装在 ContentInfo 结构中，其 ASN.1 描述如下：

```
ContentInfo{
    contentType        id-ct-scvp-certValResponse,--(1.2.840.113549.1.9.16.1.11)
    content            CVResponse}
```

如果客户端请求中不要求对响应予以保护，即 protectResponse[参见 7.1.2.3e)]的值为 FALSE，则 ContentInfo 的 content 字段就是 CVResponse 结构。

如果客户端请求没有设置 protectResponse 为 FALSE，则服务器应对响应予以保护。此时 content 字段是 SignedData 结构或 AuthenticatedData 结构，其中封装了 CVResponse。这两类结构的 ASN.1 描述参见 RFC3852。在这种情况下，SignedData 或 AuthenticatedData 的 EncapsulatedContentInfo 字段的内容如下：

——eContentType 值为 id-ct-scvp-certValResponse；

——eContent 值为 DER 编码后的 CVResponse。

CVResponse 的 ASN.1 描述如下，其中所有标签均为隐式标签。

```
CVResponse::=SEQUENCE{
    cvResponseVersion                INTEGER,
    serverConfigurationID            INTEGER,
    producedAt                       GeneralizedTime,
    responseStatus                   ResponseStatus,
    respValidationPolicy     [0]     RespValidationPolicy OPTIONAL,
    requestRef               [1]     RequestReference OPTIONAL,
    requestorRef             [2]     GeneralNames OPTIONAL,
    requestorName            [3]     GeneralNames OPTIONAL,
    replyObjects             [4]     ReplyObjects OPTIONAL,
    respNonce                [5]     OCTET STRING OPTIONAL,
    serverContextInfo        [6]     OCTET STRING OPTIONAL,
    cvResponseExtensions     [7]     Extensions OPTIONAL,
    requestorText            [8]     UTF8String (SIZE (1..256)) OPTIONAL}
```

CVResponse 的各字段含义由以下 7.1.3.2～7.1.3.14 描述。

7.1.3.2 cvResponseVersion

表示响应消息的版本号，与相应的客户端请求消息中的版本号匹配。若服务器无法产生与请求中

的版本号相应的版本号，服务器端返回一个错误信息，并说明服务器所支持的最大版本号。

7.1.3.3 serverConfigurationID

表示服务器端响应客户端的请求时服务器策略配置的版本号，详见 7.2.3.5。

7.1.3.4 producedAt

表示服务器端产生响应消息的日期及时间，应以 UTC 时间表示，具体格式参见 7.1.2.3g)中 validationTime 的定义。

7.1.3.5 responseStatus

表示服务器端对客户端请求的处理状态，其类型 ResponseStatus 的 ASN.1 描述如下：

```
ResponseStatus::=SEQUENCE{
    statusCode              CVStatusCode DEFAULT  okay,
    errorMessage            UTF8String OPTIONAL}
```

其中两个字段的含义由以下 a)和 b)描述。

a) statusCode

表示服务器端对客户端请求处理的状态，其类型 CVStatusCode 的 ASN.1 描述如下：

```
CVStatusCode::=ENUMERATED{
    okay                                (0),
    skipUnrecognizedItems               (1),
    tooBusy                             (10),
    invalidRequest                      (11),
    internalError                       (12),
    badStructure                        (20),
    unsupportedVersion                  (21),
    abortUnrecognizedItems              (22),
    unrecognizedSigKey                  (23),
    badSignatureOrMAC                   (24),
    unableToDecode                      (25),
    notAuthorized                       (26),
    unsupportedChecks                   (27),
    unsupportedWantBacks                (28),
    unsupportedSignatureOrMAC           (29),
    invalidSignatureOrMAC               (30),
    protectedResponseUnsupported        (31),
    unrecognizedResponderName           (32),
    relayingLoop                        (40),
    unrecognizedValPol                  (50),
    unrecognizedValAlg                  (51),
    fullRequestInResponseUnsupported    (52),
    fullPolResponseUnsupported          (53),
    inhibitPolicyMappingUnsupported     (54),
    requireExplicitPolicyUnsupported    (55),
```

```
    inhibitAnyPolicyUnsupported          (56),
    validationTimeUnsupported            (57),
    unrecognizedCritQueryExt             (63),
    unrecognizedCritRequestExt           (64)}
```

CVStatusCode 中每个状态值的含义如下：

——okay　客户端请求被完全处理；

——skipUnrecognizedItems　忽略了客户端请求消息中无法识别的扩展部分；

——tooBusy　服务器端繁忙；

——invalidRequest　客户端请求能够被服务器解码，仍存在其他问题；

——internalError　服务器端内部错误；

——badStructure　客户端请求消息结构错误；

——unsupportedVersion　服务器端不支持客户端请求消息的版本；

——abortUnrecognizedItems　服务器端不能处理客户端请求消息中无法识别的部分而退出；

——unrecognizedSigKey　服务器端无法验证用来保护请求消息的密钥；

——badSignatureOrMAC　签名或者 MAC 与消息体本身不符；

——unableToDecode　请求消息编码错误；

——notAuthorized　未经授权的请求消息；

——unsupportedChecks　服务器端不支持客户端请求消息中 checks 指定的处理；

——unsupportedWantBacks　服务器端不支持客户端请求消息中的 wantBack 类型；

——unsupportedSignatureOrMAC　服务器端不支持客户端请求消息中的签名或者 MAC 算法；

——invalidSignatureOrMAC　服务器端无法验证客户端请求消息中的签名或者 MAC；

——protectedResponseUnsupported　服务器端无法产生客户端请求的被保护的响应消息；

——unrecognizedResponderName　服务器端没有与被请求的响应者名称相符的证书；

——relayingLoop　客户端请求消息被同一个服务器重复传递；

——unrecognizedValPol　服务器端无法识别客户端请求中的策略引用；

——unrecognizedValAlg　服务器端无法识别客户端请求中的验证算法 OID；

——fullRequestInResponseUnsupported　服务器端不支持响应消息中包含完全的请求消息；

——fullPolResponseUnsupported　服务器端不支持响应消息中包含完全的验证策略；

——inhibitPolicyMappingUnsupported　服务器端不支持客户端请求消息中对于策略映射的限制；

——requireExplicitPolicyUnsupported　服务器端不支持客户端请求消息中显式请求的策略；

——inhibitAnyPolicyUnsupported　服务器端不支持客户端请求消息中对于允许任意策略的限制；

——validationTimeUnsupported　服务器端仅能处理使用当前时间的请求；

——unrecognizedCritQueryExt　服务器端无法识别客户端请求消息中的 query 关键扩展；

——unrecognizedCritRequestExt　服务器端无法识别客户端请求消息中的关键扩展。

状态值 0～9 指示客户端请求已被服务器处理，因此在成功处理后的响应消息中 statusCode 的值应是 0～9。如果服务器在处理中产生错误，则响应中包含错误信息且 statusCode 的值应是 10 以上。

b)　errorMessage

表示错误信息。

7.1.3.6 respValidationPolicy

respValidationPolicy 字段可选，表示服务器端验证客户端请求时使用的策略或者对策略的引用。respValidationPolicy 返回的内容是策略本身还是对策略的引用，由客户端请求的 responseValidationPolByRef 的值决定。respValidationPolicy 的类型 RespValidationPolicy 的 ASN.1 描述如下：

```
RespValidationPolicy::=ValidationPolicy
```

ValidationPolicy 的格式参见 7.1.2.3d)中定义。请求被成功处理的服务器端响应中应包含 respValidationPolicy，返回错误信息的响应中一定不能包含 respValidationPolicy。

若客户端请求中的 responseValidationPolByRef 的值为“FALSE”，ValidationPolicy 中的每一项都应被设置，若客户端请求中的 responseValidationPolByRef 的值为“TRUE”，仅当客户端请求中的值与被引用的策略不同时，ValidationPolicy 中的可选项需要被设置。

客户端应能够处理使用策略引用的服务器端响应。代理服务器应能够使用引用来声明一个策略，且应能够包含可选项。

7.1.3.7 requestRef

requestRef 字段可选。表示服务器端响应消息中对相应的请求消息的引用，其类型 RequestReference 的 ASN.1 描述如下：

```
RequestReference::=CHOICE{
    requestHash                [0]  HashValue,
    fullRequest                [1]  CVRequest}
```

其中两个字段的含义由以下 a)和 b)描述。

a) requestHash

表示对请求消息(CVRequest)的散列值，其类型 HashValue 的 ASN.1 描述如下：

```
HashValue::=SEQUENCE{
    algorithm                  AlgorithmIdentifier DEFAULT {algorithm sha-1},
    value                      OCTET STRING}
sha-1 OBJECT IDENTIFIER::={
    iso (1) identified-organization (3) oiw (14) secsig (3) algorithm (2) 26}
```

其中，algorithm 字段表示散列算法标识；value 字段表示散列值。

b) fullRequest

表示完整的请求消息，其类型为 CVRequest(定义见 7.1.2)。

客户端及代理服务器都应支持 requestHash。客户端可以支持 fullRequest，代理服务器应支持 fullRequest。

7.1.3.8 reuqestorRef

requestorRef 字段可选，表示对发起代理服务请求的原始请求者，即客户端的引用。若客户端在请求中包含了 requestorRef，代理服务器产生一个非缓存响应时，应在响应中包含与请求相同的 requestorRef 值。

7.1.3.9 requestorName

requestorName 字段可选。表示与该服务器端响应消息对应的请求消息的请求者名称。代理服务器可以在响应中包含以下几种类型的信息作为请求者名称：

a) 客户端在请求中的 requestorName 项中提供的身份信息。

b) 从证书中获得的经认证的用户身份或是用于认证请求的其他凭证。

c) 带外机制使用的用户标识。

客户端应能够处理含有 requestorName 的服务器响应。

7.1.3.10 replyObjects

replyObjects 字段可选,表示服务器端响应客户端请求的各种对象信息,每个对象说明对应请求中的一个证书的信息,其类型 ReplyObjects 的 ASN.1 描述如下,其中所有标签均为隐式标签。

```
ReplyObjects::=SEQUENCE SIZE (1..MAX) OF CertReply
CertReply::=SEQUENCE{
    cert                    CertReference,
    replyStatus             ReplyStatus DEFAULT success,
    replyValTime            GeneralizedTime,
    replyChecks             ReplyChecks,
    replyWantBacks          ReplyWantBacks,
    validationErrors   [0]  SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER
                            OPTIONAL,
    nextUpdate         [1]  GeneralizedTime OPTIONAL,
    certReplyExtensions[2]  Extensions OPTIONAL}
```

CertReply 类型中各字段的含义由以下 a)~h)描述。

a) cert

表示客户端请求消息中的证书或者证书的引用。若在客户端请求中使用了证书引用,同时请求中的 wantBack 包含 id-swb-pkc-cert 或 id-swb-aa-cert,且服务器可以获得被引用的证书,那么 cert 中应包含该证书。如果客户端在请求中使用了证书本身,则 cert 应使用请求中的 queriedCerts 的值。证书引用的类型为 CertReference,其 ASN.1 描述如下:

```
CertReference::=CHOICE{
        pkc             PKCReference,
        ac              ACReference}
```

b) replyStatus

表示服务器端返回的客户端请求消息中指定证书的状态信息,其类型 ReplyStatus 的 ASN.1 描述如下:

```
ReplyStatus::=ENUMERATED{
  success                   (0),
  malformedPKC              (1),
  malformedAC               (2),
  unavailableValidationTime (3),
  referenceCertHashFail     (4),
  certPathConstructFail     (5),
  certPathNotValid          (6),
  certPathNotValidNow       (7),
  wantBackUnsatisfied       (8)}
```

ReplyStatus 中各项值的含义如下:

——success 所有处理都成功执行;

——malformedPKC 公钥证书格式错误;

——malformedAC　　属性证书格式错误；
——unavailableValidationTime　　请求消息中的验证时间无法获得；
——referenceCertHashFail　　服务器端无法找到引用的证书或者对引用证书的散列值与请求消息中提供的不符；
——certPathConstructFail　　服务器端无法构造证书路径；
——certPathNotValid　　服务器端依照策略对证书验证的结果是无效；
——certPathNotValidNow　　当前时间下证书路径无效；
——wantBackUnsatisfied　　服务器端不能满足客户端请求消息中对回复信息的要求。

c)　replyValTime

表示服务器是在什么时间构造的 CertReply 中信息。

在客户端的请求中，validationTime 说明了用户希望服务器在什么时间执行处理。若请求中不存在 validationTime，代理服务器假设客户端指定的时间为服务器处理请求的时间，以此对请求进行响应。

d)　replyChecks

表示服务器端对客户端请求的处理类型返回的响应信息，其类型 ReplyChecks 的 ASN.1 描述如下：

```
ReplyChecks::=SEQUENCE OF ReplyCheck
ReplyCheck::=SEQUENCE{
  check          OBJECT IDENTIFIER,
  status         INTEGER DEFAULT 0}
```

ReplyCheck 中各字段的含义如下：

——check，用 OID 表示进行了哪些处理；
——status，表示处理结果状态，即是否成功。

状态值设置的具体说明如下：

对一个信任锚构造公钥证书路径({id-stc 1})，状态值为：
——0：成功构造了一个路径；
——1：无法构造路径。

对一个信任锚构造公钥证书路径并执行简单的验证({id-stc 2})，状态值为：
——0：验证有效；
——1：验证无效。

对一个信任锚构造公钥证书路径并执行完全的验证({id-stc 3})，状态值为：
——0：验证有效；
——1：验证无效；
——2：撤销信息离线，表示具有撤销信息的服务器或发布点被成功连接，但是没有数据返回，或是返回的数据不是最新的信息；
——3：撤销信息无法获得，表示连接服务器或发布点时发生网络错误，无法正常连接；
——4：撤销信息资源未知，表示服务器可成功的建立一个有效的证书路径，但是却无法对路径中的一个或多个证书给出相应的撤销信息资源。

对一个信任锚建立 AC 颁发者证书路径({id-stc 4})，状态值为：
——0：成功构造了一个路径；
——1：无法构造路径。

对一个信任锚建立 AC 颁发者证书路径并执行简单的验证({id-stc 5})，状态值为：
——0：验证有效；

——1:验证无效。

对一个信任锚建立 AC 颁发者证书路径并执行完全的验证({id-stc 6}),状态值为:

——0:验证有效;

——1:验证无效;

——2:撤销信息离线;

——3:撤销信息无法获得;

——4:撤销信息资源未知。

对一个可信根建立 AC 颁发者证书路径、执行复杂的验证并执行撤销状态检查({id-stc 7}),状态值为:

——0:验证有效;

——1:验证无效;

——2:撤销信息离线;

——3:撤销信息无法获得;

——4:撤销信息资源未知。

e) replyWantBacks

表示服务器端对客户端请求消息中的 wantBacks 的响应结果,其类型 ReplyWantBacks 的 ASN.1 描述如下:

```
ReplyWantBacks::=SEQUENCE OF ReplyWantBack
ReplyWantBack::=SEQUENCE{
  wb              OBJECT IDENTIFIER,
  value           OCTET STRING}
```

ReplyWantBack 中各字段的含义如下:

其中,wb 字段表示返回内容的 OID[参见 7.1.2.3c)],value 字段表示返回内容值。不同 OID 所对应的值的内容说明如下:

——OID:id-swb 1,value 值为 CertBundle 类型[参见 7.1.2.3h)中定义]。CertBundle 包含路径中的全部证书,从终端证书开始,以信任锚为截止。

——OID:id-swb 2,value 值为 RevInfoWantBack 类型,其 ASN.1 描述如下:RevInfoWantBack::=SEQUENCE{

```
revocationInfo          RevocationInfos,
extraCerts              CertBundle OPTIONAL}
```

若任一个被请求验证撤销信息的证书没有在 id-swb-pkc-best-cert-path 或 id-swb-pkc-all-cert-paths wantBack 中被返回,CertBundle 应在响应中存在。CertBundle 应包含全部未被返回的证书,无顺序要求。

——OID:id-swb 4,value 值为 SubjectPublicKeyInfo 类型。

——OID:id-swb 5,value 值为 CertBundle 类型,CertBundle 包含路径中的全部证书,从 AC 颁发者证书开始,以信任锚颁发的证书截止。

——OID:id-swb 6,value 值为 RevInfoWantBack 类型,若任一个被请求验证撤销信息的证书没有在 id-aa-cert-path 中被返回,CertBundle 应在响应中存在。CertBundle 应包含全部未被返回的证书,无顺序要求。

——OID:id-swb 7,value 值为 RevInfoWantBack 类型,若任一个被请求验证撤销信息的证书没有在 id-aa-cert-path 中被返回,CertBundle 应在响应中存在。CertBundle 应包含全部未被返回的证书,无顺序要求。

——OID:id-swb 12,value 值为 CertBundles 类型,每个 CertBundle 包含一个路径中的全部证书,

从终端实体证书开始，以信任锚颁发的证书截止。

——OID：id-swb 9，value 值为 SCVPResponses 类型，若代理服务器利用从其他服务器获得的信息创建响应，SCVPResponses 中应包含每一个从其他服务器获得的响应。若代理服务器产生响应时未使用任何从其他服务器获得的信息，SCVPResponses 应是一个空序列。SCVPResponses 的 ASN.1 描述如下：

SCVPResponses::=SEQUENCE OF ContentInfo

——OID：id-swb 13，value 值为 RevInfoWantBack 类型。若任一个被请求验证撤销信息的证书没有在 id-swb-pkc-best-cert-path 或 id-swb-pkc-all-cert-paths wantBack 中被返回，CertBundle 应在响应中存在。CertBundle 应包含全部未被返回的证书，无顺序要求。

——OID：id-swb 14，value 值为 RevInfoWantBack 类型。若任一个被请求验证撤销信息的证书没有在 id-swb-pkc-best-cert-path 或 id-swb-pkc-all-cert-paths wantBack 中被返回，CertBundle 应在响应中存在。CertBundle 应包含全部未被返回的证书，无顺序要求。

f) validationErrors

validationErrors 字段可选。表示服务器端在处理客户端请求时发生的错误，由一个或多个 OID 构成，每个 OID 说明一个具体的原因。validationErrors 仅在返回错误信息的响应中存在，应该仅在 replyStatus 为 3、5、6、7 或 8 的响应中出现。代理服务器不需要在响应中说明全部的验证状态失败的原因。客户端一定不能假设 validationErrors 中的 OIDs 说明了验证失败的全部原因。

g) nextUpdate

nextUpdate 字段可选。表示服务器端告知客户端什么时候可以获得更新的证书有效性信息。

h) certReplyExtensions

certReplyExtensions 字段可选。表示服务器端对客户端请求消息中 queryExtensions 的响应，其类型 Extensions 的 ASN.1 描述参见 7.1.2.3k)中定义。

certReplyExtensions 中的 extnID 应与请求中 queryExtensions 的 OID 一致，critical 应设为“FALSE”。

7.1.3.11 respNonce

respNonce 字段可选，表示服务器端响应随机数，与相应的客户端请求消息中的随机数(requestNonce)一致。若客户端请求中包含 requestNonce，且服务器对请求产生非缓存的响应，那么服务器应在响应中返回与 requestNonce 值相同的 respNonce。若服务器对请求返回缓存的响应，则响应中一定不含有 respNonce 项。

代理服务器应支持 requestNonce，客户端应能处理含有 requestNonce 的响应。

7.1.3.12 serverContextInfo

serverContextInfo 字段可选，表示服务器端向客户端传递的一些信息。这些信息对客户端来说没有明确的语义，只是用来表征一次请求/响应对的上下文环境。实际应用中若客户端期望为前一次代理路径构造请求返回一条不一样的路径，则可将前一次响应的 serverContextInfo 内容包含在新请求中[参见 7.1.2.3f)]。如果代理服务器无法为客户端返回其他的路径，则服务器端响应一定不能包含 serverContextInfo。

本标准不定义 serverContextInfo 中数据的格式。实现中可以自行定义能够表征一次请求/响应对上下文的数据，例如使用已传递给客户端的证书路径的散列值。

7.1.3.13 cvResponseExtensions

cvResponseExtensions 字段可选。表示服务器端响应消息中的扩展内容，其类型 Extensions 的定

义如 7.1.2.3k)所示。

7.1.3.14 requestorText

requestorText 字段可选。表示服务器端返回的由客户端请求提供的文本内容。若客户端请求中包含 requestorText，且服务器对请求产生非缓存的响应，服务器应在响应中返回与请求中 requestorText 值相同的 requestorText。若服务器对请求返回缓存的响应，服务器响应中一定不包含 requestorText。

客户端若能支持在请求中包含 requestorText，就应能够处理包含 requestorText 的响应。代理服务器应支持在响应中包含 requestorText。

7.2 策略配置请求/响应消息

7.2.1 概述

在代理验证和代理路径构造服务中，服务器需要按照代理验证和代理路径构造策略(参见 5.4)处理客户端请求。如果客户端不在请求消息中提供要求使用的策略，则服务器端依照自身默认的策略进行路径构造或者验证。

策略配置请求/响应用于获取代理服务器支持的策略集合。客户端可以发送策略请求信息，而服务器在收到客户端的策略请求信息后，将自身支持的策略集合和相关配置信息通过响应消息返回给客户端。

策略配置请求/响应消息应采用 GB/T 16263.1—2006 定义的 DER 编码方式进行编码。

7.2.2 客户端请求消息

7.2.2.1 客户端请求消息格式

客户端策略请求消息封装在 ValPolRequest 结构中。发送时，ValPolRequest 封装在 ContentInfo 中，其 ASN.1 描述如下：

```
ContentInfo{
    contentType          id-ct-scvp-valPolRequest,--(1.2.840.113549.1.9.16.1.12)
    content              ValPolRequest}
```

策略请求消息不需要进行保护，因此 content 的内容就是 ValPolRequest。

ValPolRequest 的 ASN.1 描述如下：

```
ValPolRequest::=SEQUENCE{
    vpRequestVersion     INTEGER DEFAULT 1,
    requestNonce         OCTET STRING}
```

ValPolRequest 类型中各字段的含义如下分别描述。

7.2.2.2 vpRequestVersion

表示策略请求消息的版本号。相应的，服务器端响应消息中的版本号应与此请求消息中的版本号一致。其语法和语义与 7.1.2.2 的描述相同。

7.2.2.3 requestNonce

表示客户端产生的随机数。如果服务器端返回一个响应消息，则应要包含客户端请求消息中的这个随机数值；如果服务器端返回的仅仅是一个缓存的消息，那么响应中一定不能包含客户端请求消息中的这个随机数值。

7.2.3 服务器端响应消息

7.2.3.1 服务器端响应消息格式

服务器端响应封装在 ValPolRequest 中，ValPolRequest 又封装在 ContentInfo 中，其 ASN.1 描述如下：

```
ContentInfo{
    contentType        id-ct-scvp-valPolResponse --(1.2.840.113549.1.9.16.1.13)
    content            SignedData}
```

服务器响应应由服务器签名，因此 content 字段的内容是 SignedData 结构，SignedData 的 EncapsulatedContentInfo 的内容如下：

——eContentType，值为 id-ct-scvp-valPolResponse；

——eContent，值为 DER 编码后的 ValPolResponse。

ValPolRequest 的 ASN.1 描述如下：

```
ValPolResponse::=SEQUENCE{
    vpResponseVersion          INTEGER,
    maxCVRequestVersion        INTEGER,
    maxVPRequestVersion        INTEGER,
    serverConfigurationID      INTEGER,
    thisUpdate                 GeneralizedTime,
    nextUpdate                 GeneralizedTime OPTIONAL,
    supportedChecks            CertChecks,
    supportedWantBacks         WantBack,
    validationPolicies         SEQUENCE OF OBJECT IDENTIFIER,
    validationAlgs             SEQUENCE OF OBJECT IDENTIFIER,
    authPolicies               SEQUENCE OF AuthPolicy,
    responseTypes              ResponseTypes,
    defaultPolicyValues        RespValidationPolicy,
    revocationInfoTypes        RevocationInfoTypes,
    signatureGeneration        SEQUENCE OF AlgorithmIdentifier,
    signatureVerification      SEQUENCE OF AlgorithmIdentifier,
    hashAlgorithms             SEQUENCE SIZE (1..MAX) OF OBJECT IDENTIFIER,
    serverPublicKeys           SEQUENCE OF KeyAgreePublicKey OPTIONAL,
    clockSkew                  INTEGER DEFAULT 10,
    requestNonce               OCTET STRING OPTIONAL}
```

ValPolResponse 中各个字段含义由以下 7.2.3.2～7.3.2.21 描述。

7.2.3.2 vpResponseVersion

表示服务器端响应客户端请求消息的版本号，与对应的客户端请求消息的版本号一致。若请求中所用版本号大于服务器支持的最大版本号，服务器端将返回其所支持的最大版本号。

7.2.3.3 maxCVRequestVersion

表示服务器端支持的证书路径构造和验证请求消息的最大版本号。

7.2.3.4 maxVPRequestVersion

表示服务器端支持的策略请求消息的最大版本号。

7.2.3.5 serverConfigurationID

表示服务器端响应客户端的请求时策略配置的版本号。当 validationPolicies,、validationAlgs、authPolicies、defaultPolicyValues 及 clockSkew 中的任何一个值发生改变时,服务器应生成一个具有新版本号的响应消息。

7.2.3.6 thisUpdate

表示服务器端签发当前策略响应消息的时间。

7.2.3.7 nextUpdate

表示服务器端更新策略响应消息的时间。服务器端响应中包含 nextUpdate 时,说明服务器返回了一个缓存的响应;若响应中不包含 nextUpdate,说明服务器对一个特定请求产生了一个非缓存的响应,服务器应按 nextupdate 定义的时间,周期性产生新的响应,服务器可能对多个不同的请求使用相同的 ValPolResponse。

7.2.3.8 supportedChecks

表示服务器端所能支持的检查类型,由一个 OID 序列构成,其类型 CertChecks 的定义如 7.1.2.3b)所示。

7.2.3.9 supportedWantBacks

表示服务器端所能支持的回复类型,由一个 OID 序列构成,其类型 WantBack 的定义如 7.1.2.3c)所示。

7.2.3.10 validationPolicies

表示服务器端支持的策略类型,由一个 OID 序列构成。若服务器不希望使用默认策略,validationPolicies 中一定不能包含 id-svp-defaultValPolicy。

7.2.3.11 validationAlgs

表示服务器端支持的验证算法类型,由一个 OID 序列构成,其中每个 OID 说明一种验证算法。

7.2.3.12 authPolicies

表示服务器端支持的鉴别策略,由一个鉴别策略引用的序列构成。其类型 AuthPolicy 的 ASN.1 描述如下:

```
AuthPolicy::=    OBJECT IDENTIFIER
```

鉴别策略的引用,是客户端与服务器端协商好的用于标识一个鉴别策略的 OID。

7.2.3.13 responseTypes

表示服务器端支持的响应类型,其类型 ResponseTypes 的 ASN.1 描述如下:

```
ResponseTypes    ::=ENUMERATED{
  cached-only                (0),
```

non-cached-only (1),
cached-and-non-cached (2)}

ResponseTypes 中各类型的含义如下：

a) cached-only：表示服务器端返回的响应消息是缓存的；

b) non-cached-only：表示服务器端返回的响应消息不是缓存的；

c) cached-and-non-cached：服务器支持缓存的及非缓存的响应，并可返回任意一种响应。

7.2.3.14 defaultPolicyValues

表示服务器端使用的默认策略，其类型 RespValidationPolicy 的定义如 7.1.3.6 所示。validationPolicy 中的全部可选项应被设置。当客户端请求引用默认策略且未对参数值进行设定时，代理服务器使用策略的默认值。

7.2.3.15 revocationInfoTypes

表示服务器端所能处理的撤销信息的类型，其类型 RevocationInfoTypes 的 ASN.1 描述如下所示：

RevocationInfoTypes::=BIT STRING{
fullCRLs (0),
deltaCRLs (1),
indirectCRLs (2),
oCSPResponses (3)}

RevocationInfoTypes 中各项的含义如下：

a) fullCRLs：表示完全证书撤销列表；

b) deltaCRLs：表示增量证书撤销列表；

c) indirectCRLs：表示间接证书撤销列表；

d) oCSPResponse 表示 OCSP 服务器响应消息。

若服务器支持某种撤销信息的类型，将 RevocationInfoTypes BIT STRING 的对应位设为 1，不支持的类型反对应的位设为 0。

7.2.3.16 signatureGeneration

表示服务器端支持的用于对 CVResponse 消息签名的签名算法类型，每个算法由一个 AlgorithmIdentifier 表示。算法列表中的第一个算法是服务器默认的签名算法。若用户未指定签名算法，服务器使用默认算法进行签名。对于没有在线私钥，且不能对 CVResponse 进行签名的服务器，signatureGeneration 为空的序列。

7.2.3.17 signatureVerification

表示服务器端支持的可验证的签名算法的类型，每个签名算法由 AlgorithmIdentifier 表示。对于不对 CVRequest 消息进行签名验证的服务器，signatureVerification 为空序列。

7.2.3.18 hashAlgorithms

表示服务器端支持的散列函数类型。算法列表中的第一个算法是服务器支持的默认算法。

7.2.3.19 severPublicKeys

表示服务器端进行密钥协商时公钥以及相应的参数，其类型 KeyAgreePublicKey 的 ASN.1 描述

如下：

```
KeyAgreePublicKey::=SEQUENCE{
  algorithm          AlgorithmIdentifier,
  publicKey          BIT STRING,
  macAlgorithm       AlgorithmIdentifier,
  kDF                AlgorithmIdentifier OPTIONAL}
```

KeyAgreePublicKey 中各字段的含义如下：

a） Algorithm：表示算法标识；

b） publicKey：表示公钥；

c） macAlgorithm：表示密钥协商后用到的对称算法；

d） kDF：表示密钥导出函数。

支持对 AuthenticatedData 代理服务请求的密钥协商的代理服务器应支持 serverPublicKeys 及 Diffie-Hellman 密钥协商算法。支持 serverPublicKeys 的代理服务器应支持 1 024 位的模指数群密钥（Modular Exponential group key）。支持 serverPublicKeys 的代理服务器还可能支持其他的 Diffie-Hellman（Diffie-Hellman groups）以及其他密钥协商算法。

7.2.3.20 clockSkew

表示服务器端允许的时钟偏移的范围，单位为分钟，默认值为 10 min。

7.2.3.21 requestNonce

表示服务器端发送的随机数。与 nextupdate 共同表示响应是否为缓存的响应消息。若响应中不存在 nextUpdate，响应中应包含 requestNonce，且值与请求中的 requestNonce 值相同。当存在 nextUpdate 时，响应中一定不能包含 requestNonce。

附 录 A
（资料性附录）
代理服务基本原理

A.1 概述

对数字证书的验证涉及许多方面，如认证路径构造、各级证书获取、CRL 获取、验证计算等。对一些存储、计算和通信能力有限的依赖方来说，这个过程非常复杂，会给系统带来沉重的计算负担和通信负担。针对这种情况，代理技术被引入到 PKI 应用中，从而显著减少依赖方进行证书认证路径构造和证书验证时的各种资源开销。

代理服务的基本原理是：依赖方将数字证书验证的部分或全部工作交由代理服务器来完成，并将执行结果反馈给依赖方。根据代理服务安全等级不同，又可分为两个级别的代理服务：

a） 代理认证路径构造；

b） 代理验证。

A.2 数字证书代理认证路径构造

对于有能力进行证书验证，只是期望以一次单一的交互收集证书认证路径信息的依赖方，将数字证书的认证路径构造交由代理服务器执行是必要的。这样做的好处是：

——通过与代理服务器的一次交互，依赖方就可以取得证书认证路径、证书撤销列表、OCSP 响应等信息的最新版本，并作为证书验证过程的输入。如果不使用证书认证路径构造代理服务器，则依赖方应自行通过 LDAP、HTTP、FTP 或其他资料库访问协议来与资料库交互以获取这些信息。

——在证书认证路径构造代理服务中，依赖方无需信任代理服务器本身，因为所获得的各种数据项都是经过相应的服务提供者签名的。

——使用代理服务器与依赖方直接访问资料库相比，效率得到了提高。依赖方向代理服务器提出的单个请求相当于依赖方向若干资料库提出的若干个请求，而且代理服务器的缓存功能减少了时延。

代理服务器为依赖方提供了统一的资料库访问接口。对不同类型的资料库，依赖方自行构造证书认证路径可能需要集成一系列的资料库访问软件以便对资料库进行访问；但通过代理服务器，资料库对依赖方是透明的，依赖方无须了解访问资料库的细节。

A.3 数字证书代理验证

对于存储、计算和通信能力有限，不能获取必要的证书和证书撤销信息来执行全部数字证书验证工作的依赖方，将数字证书的验证工作全部交由代理服务器执行是必要的。这类依赖方常出现在无线应用环境中，例如移动电话、个人数字助理（PDA）等无线终端。受到终端设备的存储、计算能力和网络通信带宽的限制，在终端本地进行证书认证路径构造和证书验证是非常困难的，会带来较大的时间延迟。对于强调时延最小化的关键应用，将证书验证交给一个可信的代理服务器来进行具有很大的好处。在这种情况下，依赖方提交目标实体证书、接受代理服务器响应，比本地进行证书认证路径构造与证书验证所需的时间要少得多。即使证书认证路径已经构造好，执行验证所需要的时间也比依赖方与代理服

务器交互所需要的时间要长，特别是在证书认证路径非常长的情况下。

另外，对于期望在其内部采用易于管理的统一证书验证策略的组织，将数字证书的验证工作全部交由代理服务器执行也是必要的。虽然依赖方有足够能力在本地进行证书验证，但由于集中管理证书验证策略的需要或者维护证书验证活动集中记录的需要，依赖方仍需要把证书验证交给一个可信的代理服务器来执行。当依赖方将证书验证交给代理服务器完成时，依赖方将如同信任本地的证书验证程序（如果有）一样信任该代理服务器。依赖方还可以指定代理服务器支持的证书验证策略，并要求代理服务器据此进行证书验证。

中 华 人 民 共 和 国
国 家 标 准
信息安全技术 数字证书代理认证路径
构造和代理验证规范
GB/T 29243—2012
*
中 国 标 准 出 版 社 出 版 发 行
北京市朝阳区和平里西街甲 2 号(100013)
北京市西城区三里河北街 16 号(100045)
网址:www.gb168.cn
服务热线:010-68522006
2013 年 5 月第一版
*
书号:155066·1-46895

GB/T 29243-2012