ICS 35.040
L 80
备案号:44630—2014

GM

中华人民共和国密码行业标准

GM/T 0029—2014

签名验签服务器技术规范

Sign and verify server technical specification

2014-02-13 发布　　2014-02-13 实施

国家密码管理局　　发布

中华人民共和国密码
行业标准
签名验签服务器技术规范
GM/T 0029—2014
*
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(100029)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)64275323 发行中心:(010)51780235
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 2.25 字数 60 千字
2014年4月第一版 2014年4月第一次印刷
*
书号:155066·2-27020 定价 41.00 元

GM/T 0029-2014

目　　次

前　言

本标准按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由密码行业标准化技术委员会提出并归口。

本标准起草单位：山东得安信息技术有限公司、成都卫士通信息产业股份公司、无锡江南信息安全工程技术中心、兴唐通信科技有限公司、上海格尔软件股份有限公司、长春吉大正元信息技术股份有限公司、上海市数字证书认证中心有限公司、北京数字认证股份有限公司、北京创原天地科技有限公司、北京三未信安科技发展有限公司、山东渔翁信息技术股份有限公司。

本标准主要起草人：刘平、孔凡玉、李元正、王妮娜、谭武征、赵丽丽、刘承、李述胜、王晓晨、高志权、宋志华。

签名验签服务器技术规范

1 范围

本标准规定了签名验签服务器的功能要求、安全要求、接口要求、检测要求和消息协议语法规范等有关内容。

本标准适用于签名验签服务器的研制设计、应用开发、管理和使用，也可用于指导签名验签服务器的检测。

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 9813 微型计算机通用规范

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

GM/T 0006—2012 密码应用标识规范

GM/T 0009 SM2 密码算法使用规范

GM/T 0010 SM2 密码算法加密签名消息语法规范

GM/T 0014 数字证书认证系统密码协议规范

GM/T 0015 基于 SM2 密码算法的数字证书格式规范

GM/T 0018 密码设备应用接口规范

GM/T 0020 证书应用综合服务接口规范

GM/T 0030 服务器密码机技术规范

PKCS #1 RSA 密码算法使用规范

PKCS #7 RSA 密码算法消息语法规范

3 术语和定义

下列术语和定义适用于本文件。

3.1

签名验签服务器 sign and verify server

用于服务端的，为应用实体提供基于 PKI 体系和数字证书的数字签名、验证签名等运算功能的服务器，可以保证关键业务信息的真实性、完整性和不可否认性。

3.2

应用实体 application entity

签名验签服务器的服务对象，可以是个人、机构或系统，其私钥存储在签名验签服务器的密码设备中，能够使用签名验签服务器进行签名及验签运算。

3.3

用户 user

与应用实体进行通信或认证的个人、机构或系统，其数字证书可导入到签名验签服务器中。

3.4

SM2 算法　SM2 algorithm

一种椭圆曲线公钥密码算法，其密钥长度为 256 比特。

3.5

SM3 算法　SM3 algorithm

一种密码杂凑算法，其输出为 256 比特。

4　缩略语

下列缩略语适用于本文件。

API　应用程序接口(Application Program Interface)

CA　证书认证机构(Certification Authority)

CRL　证书撤销列表(Certificate Revocation List)

LDAP　轻量级目录访问协议(Lightweight Directory Access Protocol)

OCSP　在线证书状态查询协议(Online Certificate Status Protocol)

PKCS　公钥密码标准(Public-Key Cryptography Standard)

PKI　公钥密码基础设施(Public Key Infrastructure)

5　签名验签服务器的功能要求

5.1　初始化功能

签名验签服务器的初始化主要包括系统配置、生成管理员等，使设备处于正常工作状态。

5.2　与 CA 基础设施的连接功能

签名验签服务器应支持与 CA 基础设施的连接功能，包括 CRL 连接配置、OCSP 连接配置等。

5.2.1　CRL 连接配置

签名验签服务器应支持 CRL 连接配置功能，通过配置管理界面，提供从 CRL 发布点获取 CRL、导入 CRL 等功能。

5.2.2　OCSP 连接配置

签名验签服务器可支持 OCSP 连接配置功能，通过配置管理界面，进行 OCSP 服务的连接配置管理。OCSP 连接配置应遵循 GB/T 19713—2005。

5.3　应用管理功能

签名验签服务器的应用管理功能主要包括应用实体的注册、配置密钥、设置私钥授权码等，并按照安全机制对应用实体的信息进行安全存储。应用实体注册的内容应包括设置应用实体名称、配置密钥索引号、导入证书、设置 IP 地址(可选)等。

5.4　证书管理和验证功能

签名验签服务器管理的证书包括应用实体证书和用户证书。签名验签服务器应具有对应用实体证书、用户证书、根证书或证书链的导入、存储、验证、使用以及备份和恢复等功能。

5.4.1 应用实体的密钥产生、证书申请

在签名验签服务器注册的应用实体，应由签名验签服务器产生应用实体的签名密钥对和证书请求，并支持通过管理界面导入应用实体的签名证书、加密证书和加密密钥对。加密密钥对的导入参见GM/T 0014。

5.4.2 用户证书导入和存储

签名验签服务器应支持用户证书、根证书或证书链的导入，导入时应对证书的有效性进行验证。

5.4.3 应用实体的证书更新

应用实体的证书更新时必须保存原来的证书，以防止以前的签名不能验证。

5.4.4 证书验证

签名验签服务器应支持对证书的有效性的验证，包括验证证书有效期、验证证书签名有效性、验证是否在 CRL 中。

5.4.5 备份/恢复

签名验签服务器应支持备份/恢复功能，包括密钥的备份恢复和证书等数据的备份/恢复。

密钥的备份恢复应通过签名验签服务器的管理界面实现。

数据的备份/恢复包括证书、配置参数等数据的备份/恢复。备份操作产生的备份文件可存储到签名验签服务器外的存储介质中，应采取措施保证备份文件的完整性；备份文件可以恢复到签名验签服务器中，同厂家的不同型号的签名验签服务器之间应能够互相备份恢复。

5.5 数字签名功能

签名验签服务器必须至少支持 SM2 算法的数字签名功能，提供对数据、消息、文件等多种格式的运算方式。

签名验签服务器可以支持 RSA 算法的数字签名功能，其中 RSA 算法模长至少为 2048 比特以上。

当签名验签服务器的公钥算法为 SM2 算法时，数据的结构遵循 GM/T 0009 或 GM/T 0010。

当签名验签服务器的公钥算法为 RSA 算法时，数据的结构遵循 PKCS#1 标准或 PKCS#7 标准。

5.6 访问控制功能

签名验签服务器的管理界面应具备完善的身份鉴别机制，通过智能密码钥匙、智能 IC 卡与口令相结合的方式实现管理员身份的鉴别，其中“口令”需要进行时效限制，超过时限，需强制修改密码。管理员登录成功后，通过管理界面进行应用管理、证书管理、系统配置以及日志查询等管理操作。

5.7 日志管理功能

签名验签服务器应提供日志记录、查看、审计和导出功能，具备相应的配置管理和查看界面。日志内容分为系统管理日志、异常事件、系统服务日志等，包括登录认证、系统配置、密钥管理等操作，认证失败、非法访问等异常事件的记录，以及与设备管理中心连接，对相应操作进行记录，对应用接口的调用进行日志记录。

5.8 系统自检功能

签名验签服务器应具备自检功能，包括自身自检和密码设备自检。签名验签服务器使用的密码设

备应具备状态和功能自检功能，能够进行密码算法正确性检查、随机数发生器检查、存储密钥和数据的完整性检查等。签名验签服务器的自身自检包括密码功能检测、存储信息的完整性检查等。

5.9 NTP 时间源同步功能

签名验签服务器能够配置时间源服务器，自动同步时间。

6 签名验签服务器的安全要求

6.1 密码设备

签名验签服务器必须使用国家密码管理主管部门审批的密码设备。调用密码设备的接口 API 应遵循 GM/T 0018。

6.2 系统要求

签名验签服务器所使用的操作系统应进行安全加固，裁减一切不需要的模块，关闭所有不需要的端口和服务。

6.3 使用要求

签名验签服务器只接受合法的操作指令，签名验签服务器软件应采用模块化设计，应通过身份鉴别等技术措施防止用户的非法调用。

6.4 管理要求

6.4.1 管理工具

签名验签服务器通过管理工具实现对该签名验签服务器的管理功能。

管理工具可以安装签名验签服务器上，也可以安装在签名验签服务器之外的管理终端上。

6.4.2 管理员管理

签名验签服务器应设置管理员和审计员，管理员和审计员的职责按照国家密码管理机构的要求进行管理。管理员和审计员应采用智能密码钥匙、智能 IC 卡等硬件装置与登录口令相结合的方式登录系统，并使用证书进行身份验证。各类管理员通过身份鉴别后执行相关管理操作。

6.4.3 设备管理

6.4.3.1 设备初始化

签名验签服务器的初始化，除必须由厂商进行的操作外，系统配置、密钥的生成(恢复)与安装、生成管理员和审计员等均应由用户方设备管理人员完成。

6.4.3.2 设备自检

签名验签服务器的自检包括密码设备的自检和自身的自检，对密码运算功能、随机数发生器和存储的敏感信息进行检查。在检查不通过时应报警并停止工作。

6.5 设备物理安全防护

签名验签服务器在工艺设计、硬件配置等方面要采取相应的保护措施，保证设备基本的物理安全防护功能。

6.6 网络部署要求

签名验签服务器应部署在局域网内，只为局域网内的应用实体和用户服务，不能为局域网外的用户使用，不能连接互联网。

建议的网络部署图如图 1～图 4 所示。

一台应用服务器对应一台签名验签服务器的网络部署图如图 1 所示。签名验签服务器只能为一台应用服务器提供服务。

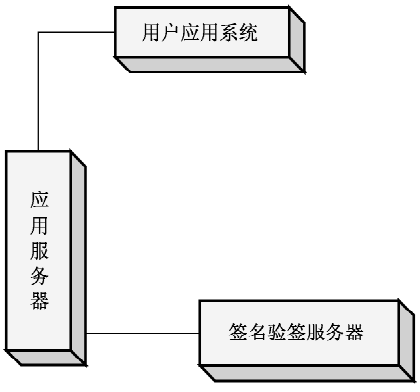

图 1 一对一

一台应用服务器对应多台签名验签服务器的网络部署图如图 2 所示。多台签名验签服务器同时为一台应用服务器提供服务。核心交换机采用双机热备。

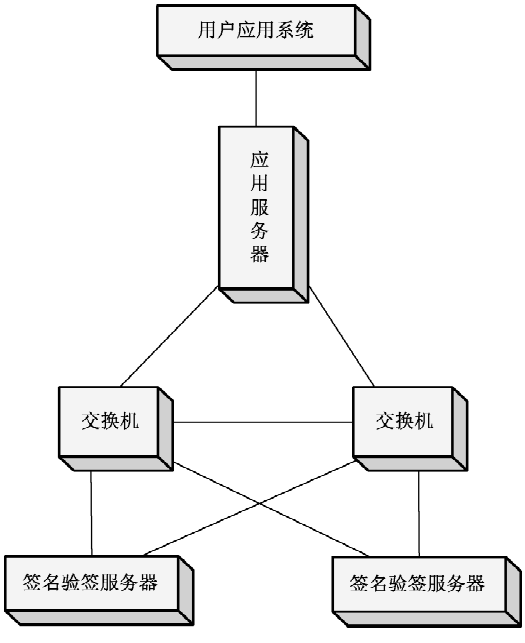

图 2 一对多

多台应用服务器对应一台签名验签服务器的网络部署图如图 3 所示。一台签名验签服务器同时为多台应用服务器提供服务。核心交换机采用双机热备。

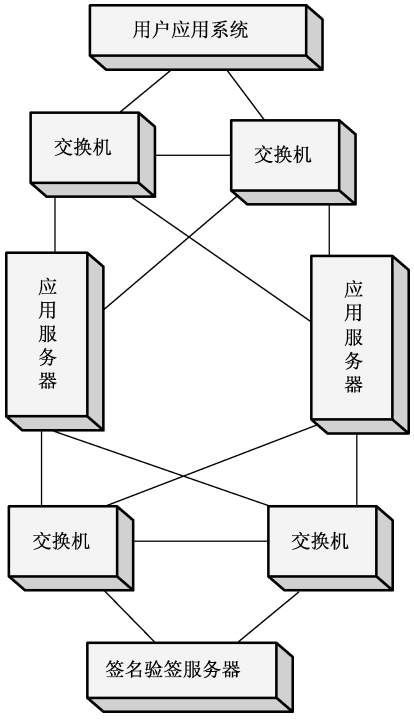

图 3　多对一

多台应用服务器对应多台签名验签服务器的网络部署图如图 4 所示。多台签名验签服务器同时为多台应用服务器提供服务。核心交换机采用双机热备。

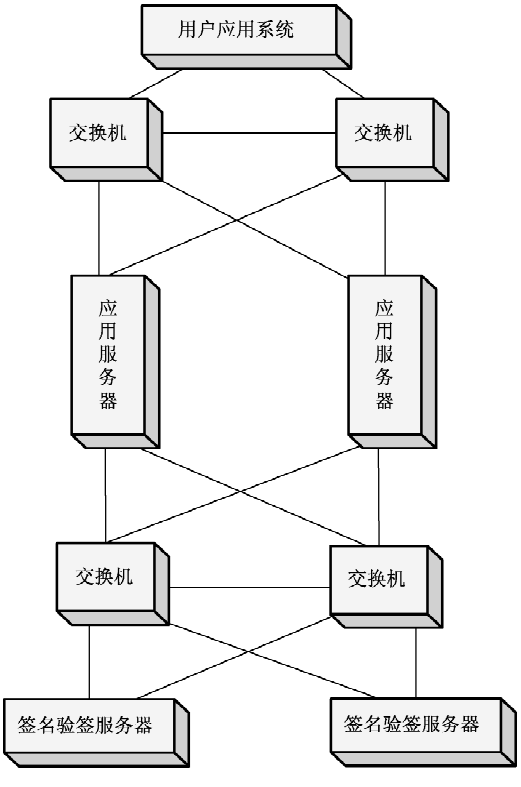

图 4　多对多

6.7 服务接口

以消息包方式提供服务的签名验签服务器，其接口应遵循附录 A 的要求，响应码的定义应遵循附录 C 的要求。

以 WEB 方式提供服务的签名验签服务器，其接口应遵循附录 B 的要求，响应码的定义应遵循附录 C 的要求。

以应用程序接口提供服务的签名验签服务器，其接口应遵循 GM/T 0020 的要求。

应用实体可以通过内部网络、USB 或者其他接口形式与签名验签服务器连接。

6.8 环境适应性

签名验签服务器的工作环境应根据实际需要遵循 GB/T 9813 中关于“气候环境适应性”的规定要求。

6.9 可靠性

签名验签服务器的平均无故障工作时间应不低于 20000 h。

7 签名验签服务器的检测要求

7.1 外观和结构的检查

根据产品的物理参数，对签名验签服务器的外观、尺寸、内部部件及附件进行检查。

7.2 提交文档的检查

签名验签服务器研制单位按照国家密码管理主管部门检测要求提交相关文档资料，作为签名验签服务器的检测依据。文档资料应包含但不限于以下内容：

a) 后台服务程序、应用编程接口和客户端管理软件的结构框图、流程图和基本功能的源代码；
b) 开机自检的工作原理说明；
c) 自测程序的工作原理说明；
d) 敏感数据信息的存储和使用说明；
e) 物理防护措施说明；
f) 技术工作总结报告；
g) 安全性设计报告；
h) 安装使用说明。

7.3 功能检测

签名验签服务器的功能检测目的是测试签名验签服务器各项功能的运行情况，并检验功能实现的正确性。

7.3.1 初始化功能检测

签名验签服务器能正常启动，对签名验签服务器进行初始化功能检测。签名验签服务器需要进行初始化检测，初始化主要包括系统配置、生成管理员，使设备处于正常工作状态。签名验签服务器应能够正常初始化，检测结果符合 5.1 和 6.4.3.1 的要求。

7.3.2 与 CA 基础设施的连接功能检测

签名验签服务器的与 CA 基础设施的连接功能检测范围包括 CRL 连接配置、OCSP 连接配置等操作，通过使用签名验签服务器的管理工具进行测试。对签名验签服务器进行与 CA 基础设施的连接功能的检测结果应符合 5.2 的要求。

7.3.3 应用管理功能检测

签名验签服务器的应用管理功能检测范围主要包括应用实体的注册和用户信息的存储，通过使用签名验签服务器的管理工具进行测试。对签名验签服务器进行应用管理功能的检测结果应符合 5.3 的要求。

7.3.4 证书管理和验证功能检测

签名验签服务器的证书管理和验证功能检测范围包括对应用实体证书、用户证书、根证书或证书链的导入、存储、验证、使用以及备份和恢复等操作，通过使用签名验签服务器的管理工具进行测试。对签名验签服务器进行证书管理和验证检测的检测结果应符合 5.4 的要求。

7.3.5 数字签名功能检测

签名验签服务器的数字签名功能测试程序由国家密码管理部门认可的检测机构设计提供。检测方法是将签名验签服务器的密码运算与第三方设备进行互通测试，如果测试互通，则测试通过；否则，测试失败。

数字签名功能检测的范围必须包括签名验签服务器提供的每个公钥密码算法的每个功能函数，如：数据、消息、文件等多种格式的运算方式，通过使用签名验签服务器的《消息协议语法规范》(见附录 A)或 GM/T 0020 进行测试。对签名验签服务器进行数字签名功能的检测结果应符合 5.5 和 6.1 的要求。

7.3.6 访问控制功能检测

通过使用签名验签服务器的管理工具或管理界面进行签名验签服务器的访问控制检测。对签名验签服务器的不同管理操作设置不同的操作权限，登录签名验签服务器应具备完善的身份鉴别机制；签名验签服务器应拒绝任何不具备相应权限的访问或操作。对签名验签服务器进行访问控制检测，检测结果应符合 5.6 的要求。

7.3.7 日志管理功能检测

通过使用签名验签服务器的日志管理工具或管理界面进行签名验签服务器的日志审计检测。签名验签服务器应提供日志记录、查看、审计和导出功能；签名验签服务器的日志内容分为系统管理日志、异常事件、系统服务日志等，包括登录认证、系统配置、密钥管理等操作，以及认证失败、非法访问等异常事件的记录。对签名验签服务器进行日志管理检测的检测结果应符合 5.7 的要求。

7.3.8 系统自检功能检测

签名验签服务器的系统自检功能主要包括密码算法正确性检查、随机数发生器检查、存储密钥和数据的完整性检查，以及关键部件的正确性检测等。对签名验签服务器进行系统自检检测的检测结果应符合 5.8 和 6.4.3.2 的要求。

7.3.9 NTP 时间源同步功能检测

对签名验签服务器进行 NTP 时间源同步的检测结果应符合 5.9 的要求。

7.3.10 服务接口检测

签名验签服务器的服务接口必须遵循附录 A、附录 B 或 GM/T 0020。对签名验签服务器进行应用编程接口检测：对于正确的调用环境和调用过程，API 函数应该返回正确的结果，并完成相应功能；对于设定的不正确的调用环境和调用过程，API 函数应返回相应的错误代码。

7.3.11 管理工具检测

通过使用签名验签服务器的管理工具或管理界面进行签名验签服务器的管理工具检测。对签名验签服务器进行管理工具检测，检测结果应符合 6.4.1 的要求。

7.4 性能检测

目的是测试签名验签服务器进行数字签名等运算的速度指标。

下列各项速度性能测试中的测试量由数据报文长度和测试次数决定。可以根据各个测试项的具体耗时情况，依照等比序列来选取测试次数，例如：测试次数 N 可以选择 1 次、10 次、100 次、1 000 次等，分别测试后得到不同测试次数时的性能序列。数据报文长度的选择在各个速度性能测试项中分别定义。

各个测试项的速度性能的计算如下式所示：

$$S = N/T$$

其中，S 为速度，单位为 tps(次/秒)；N 为测试次数；T 为测量所耗费的时间，单位为秒。

对签名验签服务器的数字签名和数字信封等功能进行性能检测的方法如下：

将一个定长数据报文，发送给签名验签服务器进行数字签名和数字信封操作，重复操作 N 次，测量其完成时间 T。用于测试的数据由检测机构选取。测试应进行多次，结果取平均值。

如签名验签服务器支持多种公钥密码算法，必须测试所支持的所有公钥密码算法及其各种应用模式。

数字签名和数字信封性能单位统一为 tps(次/秒)。

7.5 环境适应性检测

环境适应性检测应按照 GB/T 9813—2000 中 5.8 的要求进行，其结果应符合本标准 6.8 的要求。

7.6 其他检测

外观和结构检查、提交文档的检查按照相关标准进行。

8 合格判定

本标准中，除 7.3.7、7.4 以及 7.5 以外的各项检测中，其任意一项检测结果不合格，判定为产品不合格。

附 录 A
（规范性附录）
消息协议语法规范

A.1 概述

签名验签服务的消息协议接口采用请求响应模式，如图 A.1 所示。协议模型由请求者、响应者和它们之间的交互协议组成。通过本协议，请求者将数字签名、验证数字签名等请求发送给响应者，由响应者完成签名验签服务并返回结果。本规范中的接口消息协议包括导出证书、解析证书、验证证书有效性、数字签名、验证数字签名、消息签名、验证消息签名等服务功能，每个服务都按照请求—响应的步骤执行。请求者可通过本协议获得签名验签功能，而不必关心下层 PKI 公钥密码基础设施的实现细节。

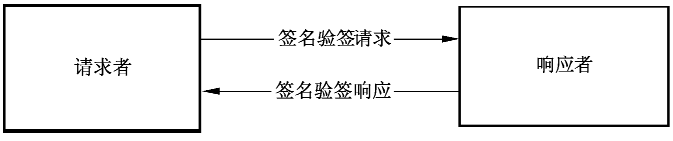

图 A.1 签名验签服务消息协议

请求者组织业务服务请求，发送到响应者，并延缓自身的事务处理过程，等待响应者响应返回；响应者接收到来自请求者的业务服务请求后，检查请求的合法性，根据请求类型处理服务请求，并将处理结果返回给请求者。

下面的协议内容将按照图 A.1 所示的框架进行。

A.2 协议内容

协议内容如下：

a） 请求：

也称业务服务请求，包含请求者业务请求的类型、性质以及特性数据等，该请求将被发送到响应者并得到服务。服务请求包括如下数据：

——协议版本（当前版本为 1）；

——请求类型；

——请求包；

——请求时间。

b） 响应：

指响应者对来自请求者请求的处理响应。响应者的响应包括如下数据：

——协议版本（当前版本为 1）；

——响应类型；

——响应包；

——响应时间。

c） 异常情况：

当响应者处理发生错误时，需要向请求者发送错误信息。错误可以是下列两类：

——请求失败：响应者验证来自请求者业务请求数据失败，请求者收到该响应后应重新组织业务请

求数据进行发送。

——内部处理失败:响应者处理请求者业务请求过程中发生内部错误,响应者通知请求者该请求处理失败,请求者需重新组织业务请求数据进行发送。

本标准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。

A.3 请求协议

A.3.1 请求数据格式

请求者请求数据的基本格式如下:

```
SVSRequest ::= SEQUENCE {
version                 Version DEFAULT v1,
reqType                 ReqType,
request                 Request,
reqTime                 GeneralizedTime
}
```

其中:

```
Version ::= INTEGER { v1(0)}
ReqType::= INTEGER{
exportCert                          (0),
parseCert                           (1),
validateCert                        (2),
signData                            (3),
verifySignedData                    (4),
signDataInit                        (5),
signDataUpdate                      (6),
signDataFinal                       (7),
verifySignedDataInit                (8),
verifySignedDataUpdate              (9),
verifySignedDataFinal               (10),
signMessage                         (11),
verifySignedMessage                 (12),
signMessageInit                     (13),
signMessageUpdate                   (14),
signMessageFinal                    (15),
verifySignedMessageInit             (16),
verifySignedMessageUpdate           (17),
verifySignedMessageFinal            (18)
}
Request ::=CHOICE{
exportUserCertReq               [0] IMPLICT ExportUserCert Req,
parseCertReq                    [1] IMPLICT ParseCertReq,
validateCertReq                 [2] IMPLICT ValidateCertReq,
```

```
    signDataReq                     [3] IMPLICT SignDataReq,
    verifySignedDataReq             [4] IMPLICT VerifySignedDataReq,
    signDataInitReq                 [5] IMPLICT SignDataInitReq,
    signDataUpdateReq               [6] IMPLICT SignDataUpdateReq,
    signDataFinalReq                [7] IMPLICT SignDataFinalReq,
    verifySignedDataInitReq         [8] IMPLICT VerifySignedDataInitReq,
    verifySignedDataUpdateReq       [9] IMPLICT VerifySignedDataUpdateReq,
    verifySignedDataFinalReq        [10] IMPLICT VerifySignedDataFinalReq,
    signMessageReq                  [11] IMPLICT SignMessageReq,
    verifySignedMessageReq          [12] IMPLICT VerifySignedMessageReq,
    signMessageInitReq              [13] IMPLICT SignMessageInitReq,
    signMessageUpdateReq            [14] IMPLICT SignMessageUpdateReq,
    signMessageFinalReq             [15] IMPLICT SignMessageFinalReq,
    verifySignedMessageInitReq      [16] IMPLICT VerifySignedMessageInitReq,
    verifySignedMessageUpdateReq    [17] IMPLICT VerifySignedMessageUpdateReq,
    verifySignedMessageFinalReq     [18] IMPLICT VerifySignedMessageFinalReq
    }
```

A.3.2 SVSRequest 及其结构解释

SVSRequest 包含了请求语法中的重要信息，本条将对该结构作详细的描述和解释：

a) 协议版本：

本项描述了请求语法的版本号，当前版本为 1，取整型值 0。

b) 请求类型：

本项描述了不同业务的请求类型值。

c) 请求包：

请求包与请求类型值之间的对应关系如表 A.1 所示。

表 A.1 请求包与请求类型值的对应关系

应答类型字符描述	应答类型值	响应包说明
exportCert	0	导出证书申请包
parseCert	1	解析证书申请包
validateCert	2	验证证书有效性申请包
signData	3	单包数字签名申请包
verifySignedData	4	单包验证数字签名申请包
signDataInit	5	多包数字签名初始化申请包
signDataUpdate	6	多包数字签名更新申请包
signDataFinal	7	多包数字签名结束申请包
verifySignedDataInit	8	多包验证数字签名初始化申请包
verifySignedDataUpdate	9	多包验证数字签名更新申请包
verifySignedDataFinal	10	多包验证数字签名结束申请包

表 A.1（续）

请求类型字符描述	请求类型值	申请包说明
signMessage	11	单包消息签名申请包
verifySignedMessage	12	单包验证消息签名申请包
signMessageInit	13	多包消息签名初始化申请包
signMessageUpdate	14	多包消息签名更新申请包
signMessageFinal	15	多包消息签名结束申请包
verifySignedMessageInit	16	多包验证消息签名初始化申请包
verifySignedMessageUpdate	17	多包验证消息签名更新申请包
verifySignedMessageFinal	18	多包验证消息签名结束申请包

d） 请求时间：

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

A.4 响应协议

A.4.1 响应数据格式

响应者响应的基本格式如下：

```
SVSRespond ::= SEQUENCE {
version                 Version DEFAULT v1,
respType                RespType,
respond                 Respond,
respTime                GeneralizedTime
}
```

其中：

```
Version ::= INTEGER { v1(0)}
RespType::= INTEGER{
exportCert                      (0),
parseCert                       (1),
validateCert                    (2),
signData                        (3),
verifySignedData                (4),
signDataInit                    (5),
signDataUpdate                  (6),
signDataFinal                   (7),
verifySignedDataInit            (8),
verifySignedDataUpdate          (9),
verifySignedDataFinal           (10),
signMessage                     (11),
```

```
verifySignedMessage              (12),
signMessageInit                  (13),
signMessageUpdate                (14),
signMessageFinal                 (15),
verifySignedMessageInit          (16),
verifySignedMessageUpdate        (17),
verifySignedMessageFinal         (18)
}
Respond::=CHOICE{
exportUserCertResp                    [0] IMPLICT ExportUserCert Resp,
parseCertResp                         [1] IMPLICT ParseCertResp,
validateCertResp                      [2] IMPLICT ValidateCertResp,
signDataResp                          [3] IMPLICT SignDataResp,
verifySignedDataResp                  [4] IMPLICT VerifySignedDataResp,
signDataInitResp                      [5] IMPLICT SignDataInitResp,
signDataUpdateResp                    [6] IMPLICT SignDataUpdateResp,
signDataFinalResp                     [7] IMPLICT SignDataFinalResp,
verifySignedDataInitResp              [8] IMPLICT VerifySignedDataInitResp,
verifySignedDataUpdateResp            [9] IMPLICT VerifySignedDataUpdateResp,
verifySignedDataFinalResp             [10] IMPLICT VerifySignedDataFinalResp,
signMessageResp                       [11] IMPLICT SignMessageResp,
verifySignedMessageResp               [12] IMPLICT VerifySignedMessageResp,
signMessageInitResp                   [13] IMPLICT SignMessageInitResp,
signMessageUpdateResp                 [14] IMPLICT SignMessageUpdateResp,
signMessageFinalResp                  [15] IMPLICT SignMessageFinalResp,
verifySignedMessageInitResp           [16] IMPLICT VerifySignedMessageInitResp,
verifySignedMessageUpdateResp         [17] IMPLICT VerifySignedMessageUpdateResp,
verifySignedMessageFinalResp          [18] IMPLICT VerifySignedMessageFinalResp
}
```

A.4.2 SVSRespond 及其结构解释

SVSRespond 包含了响应语法中的重要信息，本条将对该结构作详细的描述和解释：

a) 协议版本：

本项描述了响应语法的版本号，当前版本为 1，取整型值 0。

b) 响应类型：

本项描述了不同业务的响应类型值。

c) 响应包：

响应包与响应类型值之间的对应关系如表 A.2 所示。

表 A.2　响应包与相应类型值的对应关系

应答类型字符描述	应答类型值	响应包说明
exportCert	0	导出证书响应包
parseCert	1	解析证书响应包
validateCert	2	验证证书有效性响应包
signData	3	单包数字签名响应包
verifySignedData	4	单包验证数字签名响应包
signDataInit	5	多包数字签名初始化响应包
signDataUpdate	6	多包数字签名更新响应包
signDataFinal	7	多包数字签名结束响应包
verifySignedDataInit	8	多包验证数字签名初始化响应包
verifySignedDataUpdate	9	多包验证数字签名更新响应包
verifySignedDataFinal	10	多包验证数字签名结束响应包
signMessage	11	单包消息签名响应包
verifySignedMessage	12	单包验证消息签名响应包
signMessageInit	13	多包消息签名初始化响应包
signMessageUpdate	14	多包消息签名更新响应包
signMessageFinal	15	多包消息签名结束响应包
verifySignedMessageInit	16	多包验证消息签名初始化响应包
verifySignedMessageUpdate	17	多包验证消息签名更新响应包
verifySignedMessageFinal	18	多包验证消息签名结束响应包

d)　响应时间：

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

A.5　协议接口功能说明

A.5.1　导出证书

——ExportCertReq 包：

ExportCertReq 包为导出证书请求格式包，当 reqType 取值 exportCert 时，请求包采用本子包，其具体格式如下：

```
ExportCertReq::= SEQUENCE {
identification                OCTET STRING
}
```

identification 表明要导出证书的标识。

——ExportCertResp 包：

ExportCertResp 包为导出证书响应格式包，当 respType 取值 exportCert 时，响应包采用本子包，其具体格式如下：

```
ExportCertResp::= SEQUENCE {
```

```
respValue                INTEGER,
cert                     Certificate OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

cert 表明导出的证书。

A.5.2 解析证书

——ParseCertReq 包：

ParseCertReq 包为解析证书请求格式包，当 reqType 取值 parseCert 时，请求包采用本子包，其具体格式如下：

```
ParseCertReq::= SEQUENCE {
infoType                 INTEGER,
cert                     Certificate
}
```

infoType 表明要解析证书信息的类型，详细定义见 GM/T 0006—2012 中 6.3.4 证书解析项标识；

cert 表示要解析的数字证书。

——ParseCertResp 包：

ParseCertResp 包为解析证书响应格式包，当 respType 取值 parseCert 时，响应包采用本子包，其具体格式如下：

```
ParseCertResp::= SEQUENCE {
respValue                INTEGER,
info                     OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

info 表示获取的证书信息。

A.5.3 验证证书有效性

——ValidateCertReq 包：

ValidateCertReq 包为验证证书有效性请求格式包，当 reqType 取值 validateCert 时，请求包采用本子包，其具体格式如下：

```
ValidateCertReq::= SEQUENCE {
cert                     Certificate,
ocsp                     BOOLEAN DEFAULT FALSE
}
```

cert 表示要验证证书有效性的数字证书；

ocsp 表示是否获取证书 OCSP 状态，默认值为 FALSE。

——ValidateCertResp 包：

ValidateCertResp 包为验证证书有效性响应格式包，当 respType 取值 validateCert 时，响应包采用本子包，其具体格式如下：

```
ValidateCertResp::= SEQUENCE {
respValue                INTEGER,
state                    INTEGER OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

state 表明获取的证书 OCSP 状态标识。

A.5.4 单包数字签名

——SignDataReq 包：

SignDataReq 包为单包数字签名请求格式包，当 reqType 取值 signData 时，请求包采用本子包，其具体格式如下：

```
SignDataReq::= SEQUENCE {
signMethod              INTEGER,
keyIndex                INTEGER,
keyValue                OCTET STRING,
inDataLen               INTEGER,
inData                  OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

keyIndex 表示签名者私钥的索引值，如十进制 1 表示索引值为 1 的密钥；

keyValue 表示签名者私钥权限标识码；

inDataLen 表示待签名的数据原文长度；

inData 表示待签名的数据原文。

——SignDataResp 包：

SignDataResp 包为单包数字签名响应格式包，当 respType 取值 signData 时，响应包采用本子包，其具体格式如下：

```
SignDataResp::= SEQUENCE {
respValue               INTEGER,
signature               OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

signature 表示签名值，当公钥算法为 RSA 时，数据的结构遵循 PKCS#1；当公钥算法为 SM2 时，数据的结构遵循 GM/T 0009。

A.5.5 单包验证数字签名

——VerifySignedDataReq 包：

VerifySignedDataReq 包为单包验证数字签名请求格式包，当 reqType 取值 verifySignedData 时，请求包采用本子包，其具体格式如下：

```
VerifySignedDataReq::= SEQUENCE {
type                    INTEGER,
cert                    [0] IMPLICT Certificate OPTIONAL,
certSN                  [1] IMPLICT OCTET STRING OPTIONAL,
inDataLen               INTEGER,
inData                  OCTET STRING,
signature               OCTET STRING,
verifyLevel             INTEGER
}
```

type 表示使用验证数字签名时使用证书或证书序列号，1 表示使用证书，2 表示使用证书序列号；

cert 表示签名证书，type 取值 1 时有效；

certSN 表示签名证书序列号，type 取值 2 时有效；

inDataLen 表示待签名的数据原文长度；

inData 表示待签名的数据原文；

signature 表示签名值，当公钥算法为 RSA 时，数据的结构遵循 PKCS＃1；当公钥算法为 SM2 时，数据的结构遵循 GM/T 0009；

verifyLevel 表示证书验证级别，0：验证时间，1：验证时间和根证书签名，2：验证时间、根证书签名和 CRL。

——VerifySignedDataResp 包：

VerifySignedDataResp 包为单包验证数字签名响应格式包，当 respType 取值 verifySignedData 时，响应包采用本子包，其具体格式如下：

```
VerifySignedDataResp::= SEQUENCE {
respValue                INTEGER
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

A.5.6 多包数字签名初始化

——SignDataInitReq 包：

SignDataInitReq 包为多包数字签名初始化请求格式包，当 reqType 取值 signDataInit 时，请求包采用本子包，其具体格式如下：

```
SignDataInitReq::= SEQUENCE {
signMethod               INTEGER,
signerPublicKey          [0] IMPLICT OCTET STRING OPTIONAL,
signerIDLen              [1] IMPLICT INTEGER OPTIONAL,
signerID                 [2] IMPLICT OCTET STRING OPTIONAL,
inDataLen                INTEGER,
inData                   OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

signerPublicKey 表示签名者公钥，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerIDLen 表示签名者的 ID 长度，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerID 表示签名者的 ID 值，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——SignDataInitResp 包：

SignDataInitResp 包为多包数字签名初始化响应格式包，当 respType 取值 signDataInit 时，响应包采用本子包，其具体格式如下：

```
SignDataInitResp::= SEQUENCE {
respValue                INTEGER,
hashValue                OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.7 多包数字签名更新

——SignDataUpdateReq 包：

SignDataInitReq 包为多包数字签名更新请求格式包，当 reqType 取值 signDataUpdate 时，请求包采用本子包，其具体格式如下：

```
SignDataUpdateReq::= SEQUENCE {
signMethod              INTEGER,
hashValueLen            INTEGER,
hashValue               OCTET STRING,
inDataLen               INTEGER,
inData                  OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——SignDataUpdateResp 包：

SignDataUpdateResp 包为多包数字签名更新响应格式包，当 respType 取值 signDataUpdate 时，响应包采用本子包，其具体格式如下：

```
SignDataUpdateResp::= SEQUENCE {
respValue               INTEGER,
hashValue               OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.8 多包数字签名结束

——SignDataFinalReq 包：

SignDataReq 包为多包数字签名结束请求格式包，当 reqType 取值 signDataFinal 时，请求包采用本子包，其具体格式如下：

```
SignDataFinalReq::= SEQUENCE {
signMethod              INTEGER,
keyIndex                INTEGER,
keyValue                OCTET STRING,
hashValueLen            INTEGER,
hashValue               OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

keyIndex 表示签名者私钥的索引值，如十进制 1 表示索引值为 1 的密钥；

keyValue 表示签名者私钥权限标识码；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值。

——SignDataFinalResp 包：

SignDataResp 包为多包数字签名结束响应格式包，当 respType 取值 signDataFinal 时，响应包采用本子包，其具体格式如下：

```
SignDataFinalResp::= SEQUENCE {
respValue                INTEGER,
signature                OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

signature 表示签名值，当公钥算法为 RSA 时，数据的结构遵循 PKCS#1；当公钥算法为 SM2 时，数据的结构遵循 GM/T 0009。

A.5.9 多包验证数字签名初始化

——VerifySignedDataInitReq 包：

VerifySignedDataInitReq 包为多包验证数字签名初始化请求格式包，当 reqType 取值 verifySignedDataInit 时，请求包采用本子包，其具体格式如下：

```
VerifySignedDataInitReq::= SEQUENCE {
signMethod               INTEGER,
signerPublicKey          [0] IMPLICT OCTET STRING OPTIONAL,
signerIDLen              [1] IMPLICT INTEGER OPTIONAL,
signerID                 [2] IMPLICT OCTET STRING OPTIONAL,
inDataLen                INTEGER,
inData                   OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

signerPublicKey 表示签名者公钥，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerIDLen 表示签名者的 ID 长度，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerID 表示签名者的 ID 值，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——VerifySignedDataInitResp 包：

VerifySignedDataInitResp 包为多包验证数字签名初始化响应格式包，当 respType 取值 verifySignedDataInit 时，响应包采用本子包，其具体格式如下：

```
VerifySignedDataInitResp::= SEQUENCE {
respValue                INTEGER,
hashValue                OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.10 多包验证数字签名更新

——VerifySignedDataUpdateReq 包：

VerifySignedDataUpdateReq 包为多包验证数字签名更新请求格式包，当 reqType 取值 verifySignedDataUpdate 时，请求包采用本子包，其具体格式如下：

```
VerifySignedDataUpdateReq::= SEQUENCE {
signMethod              INTEGER,
hashValueLen            INTEGER,
hashValue               OCTET STRING,
inDataLen               INTEGER,
inData                  OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——VerifySignedDataUpdateResp 包：

VerifySignedDataUpdateResp 包为多包验证数字签名更新响应格式包，当 respType 取值 verifySignedDataUpdate 时，响应包采用本子包，其具体格式如下：

```
VerifySignedDataUpdateResp::= SEQUENCE {
respValue               INTEGER,
hashValue               OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.11 多包验证数字签名结束

——VerifySignedDataFinalReq 包：

VerifySignedDataReq 包为多包验证数字签名结束请求格式包，当 reqType 取值 verifySignedDataFinal 时，请求包采用本子包，其具体格式如下：

```
VerifySignedDataFinalReq::= SEQUENCE {
signMethod              INTEGER,
type                    INTEGER,
cert                    [0] IMPLICT Certificate OPTIONAL,
certSN                  [1] IMPLICT OCTET STRING OPTIONAL,
hashValueLen            INTEGER,
hashValue               OCTET STRING,
signature               OCTET STRING,
verifyLevel             INTEGER
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

type 表示使用验证数字签名时使用证书或证书序列号，1 表示使用证书，2 表示使用证书序列号；

cert 表示签名证书，type 取值 1 时有效；

certSN 表示签名证书序列号，type 取值 2 时有效；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

signature 表示签名值，当公钥算法为 RSA 时，数据的结构遵循 PKCS#1；当公钥算法为 SM2 时，数据的结构遵循 GM/T 0009。

verifyLevel 表示证书验证级别，0：验证时间，1：验证时间和根证书签名，2：验证时间、根证书签名和 CRL。

——VerifySignedDataFinalResp 包：

VerifySignedDataFinalResp 包为多包验证数字签名结束响应格式包，当 respType 取值 verifySignedDataFinal 时，响应包采用本子包，其具体格式如下：

```
VerifySignedDataFinalResp::= SEQUENCE {
respValue                    INTEGER
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

A.5.12 单包消息签名

——SignMessageReq 包：

SignMessageReq 包为单包消息签名请求格式包，当 reqType 取值 signMessage 时，请求包采用本子包，其具体格式如下：

```
SignMessageReq::= SEQUENCE {
signMethod                   INTEGER,
keyIndex                     INTEGER,
keyValue                     OCTET STRING,
inDataLen                    INTEGER,
inData                       OCTET STRING,
hashFlag                     BOOLEAN DEFAULT FALSE,
originalText                 [0] IMPLICT BOOLEAN OPTIONAL,
certificateChain             [1] IMPLICT BOOLEAN OPTIONAL,
crl                          [2] IMPLICT BOOLEAN OPTIONAL,
authenticationAttributes     [3] IMPLICT BOOLEAN OPTIONAL
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

keyIndex 表示签名者私钥的索引值，如十进制 1 表示索引值为 1 的密钥；

keyValue 表示签名者私钥权限标识码；

inDataLen 表示待签名的数据长度；

inData 表示待签名的数据；

hashFlag 表示待签名的数据是否已哈希，默认 FALSE 表示未哈希；

originalText 表示是否附加原文选项；

certificateChain 表示是否附加证书链选项；

crl 表示是否附加黑名单选项；

authenticationAttributes 表示是否附加鉴别属性选项。

——SignMessageResp 包：

SignMessageResp 包为单包消息签名响应格式包，当 respType 取值 signMessage 时，响应包采用本子包，其具体格式如下：

```
SignMessageResp::= SEQUENCE {
```

```
respValue               INTEGER,
signedMessage           OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

signedMessage 表示消息签名数据，当公钥算法为 RSA 时，消息的结构遵循 PKCS#7；当公钥算法为 SM2 时，消息的结构遵循 GM/T 0010。

A.5.13 单包验证消息签名

——VerifySignedMessageReq 包：

VerifySignedMessageReq 包为单包验证消息签名请求格式包，当 reqType 取值 verifySignedMessage 时，请求包采用本子包，其具体格式如下：

```
VerifySignedMessageReq::= SEQUENCE {
inDataLen               INTEGER,
inData                  OCTET STRING,
signedMessage           OCTET STRING,
hashFlag                BOOLEAN DEFAULT FALSE,
originalText            [0] IMPLICT BOOLEAN OPTIONAL,
certificateChain        [1] IMPLICT BOOLEAN OPTIONAL,
crl                     [2] IMPLICT BOOLEAN OPTIONAL,
authenticationAttributes [3] IMPLICT BOOLEAN OPTIONAL
}
```

inDataLen 表示待签名的数据原文长度；

inData 表示待签名的数据原文；

signedMessage 表示输入的消息签名数据，当公钥算法为 RSA 时，消息的结构遵循 PKCS#7；当公钥算法为 SM2 时，消息的结构遵循 GM/T 0010；

hashFlag 表示待签名的数据是否已哈希，默认 FALSE 表示未哈希；

originalText 表示是否附加原文选项；

certificateChain 表示是否附加证书链选项；

crl 表示是否附加黑名单选项；

authenticationAttributes 表示是否附加鉴别属性选项。

——VerifySignedMessageResp 包：

VerifySignedMessageResp 包为单包验证消息签名响应格式包，当 respType 取值 verifySignedMessage 时，响应包采用本子包，其具体格式如下：

```
VerifySignedMessageResp::= SEQUENCE {
respValue               INTEGER
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

A.5.14 多包消息签名初始化

——SignMessageInitReq 包：

SignDataInitReq 包为多包消息签名初始化请求格式包，当 reqType 取值 signMessageInit 时，请求包采用本子包，其具体格式如下：

```
SignMessageInitReq::= SEQUENCE {
```

```
signMethod              INTEGER,
signerPublicKey         [0] IMPLICT OCTET STRING OPTIONAL,
signerIDLen             [1] IMPLICT INTEGER OPTIONAL,
signerID                [2] IMPLICT OCTET STRING OPTIONAL,
inDataLen               INTEGER,
inData                  OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

signerPublicKey 表示签名者公钥，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerIDLen 表示签名者的 ID 长度，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerID 表示签名者的 ID 值，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——SignMessageInitResp 包：

SignMessageInitResp 包为多包消息签名初始化响应格式包，当 respType 取值 signMessageInit 时，响应包采用本子包，其具体格式如下：

```
SignMessageInitResp::= SEQUENCE {
respValue               INTEGER,
hashValue               OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.15 多包消息签名更新

——SignMessageUpdateReq 包：

SignMessageInitReq 包为多包消息签名更新请求格式包，当 reqType 取值 signMessageUpdate 时，请求包采用本子包，其具体格式如下：

```
SignMessageUpdateReq::= SEQUENCE {
signMethod              INTEGER,
hashValueLen            INTEGER,
hashValue               OCTET STRING,
inDataLen               INTEGER,
inData                  OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——SignMessageUpdateResp 包：

SignMessageUpdateResp 包为多包消息签名更新响应格式包，当 respType 取值 signMessageUpdate 时，响应包采用本子包，其具体格式如下：

```
SignMessageUpdateResp::= SEQUENCE {
respValue                INTEGER,
hashValue                OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.16 多包消息签名结束

——SignMessageFinalReq 包：

SignMessageReq 包为多包消息签名结束请求格式包，当 reqType 取值 signMessageFinal 时，请求包采用本子包，其具体格式如下：

```
SignMessageFinalReq::= SEQUENCE {
signMethod               INTEGER,
keyIndex                 INTEGER,
keyValue                 OCTET STRING,
hashValueLen             INTEGER,
hashValue                OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

keyIndex 表示签名者私钥的索引值，如十进制 1 表示索引值为 1 的密钥；

keyValue 表示签名者私钥权限标识码；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值。

——SignMessageFinalResp 包：

SignMessageResp 包为多包消息签名结束响应格式包，当 respType 取值 signMessageFinal 时，响应包采用本子包，其具体格式如下：

```
SignMessageFinalResp::= SEQUENCE {
respValue                INTEGER,
signedMessage            OCTET STRING OPTIONAL
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

signedMessage 表示消息签名数据，当公钥算法为 RSA 时，消息的结构遵循 PKCS#7；当公钥算法为 SM2 时，消息的结构遵循 GM/T 0010。

A.5.17 多包验证消息签名初始化

——VerifySignedMessageInitReq 包：

VerifySignedMessageInitReq 包为多包验证消息签名初始化请求格式包，当 reqType 取值 verifySignedMessageInit 时，请求包采用本子包，其具体格式如下：

```
VerifySignedMessageInitReq::= SEQUENCE {
signMethod               INTEGER,
signerPublicKey          [0] IMPLICT OCTET STRING OPTIONAL,
signerIDLen              [1] IMPLICT INTEGER OPTIONAL,
signerID                 [2] IMPLICT OCTET STRING OPTIONAL,
```

inDataLen　　INTEGER，
inData　　OCTET STRING
}

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

signerPublicKey 表示签名者公钥，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerIDLen 表示签名者的 ID 长度，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

signerID 表示签名者的 ID 值，当 signMethod 为 SGD_SM3_SM2 或 SGD_SM3_RSA 时有效；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——VerifySignedMessageInitResp 包：

VerifySignedMessageInitResp 包为多包验证消息签名初始化响应格式包，当 respType 取值 verifySignedMessageInit 时，响应包采用本子包，其具体格式如下：

VerifySignedMessageInitResp::= SEQUENCE {
respValue　　INTEGER，
hashValue　　OCTET STRING OPTIONAL
}

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.18　多包验证消息签名更新

——VerifySignedMessageUpdateReq 包：

VerifySignedMessageUpdateReq 包为多包验证消息签名更新请求格式包，当 reqType 取值 verifySignedMessageUpdate 时，请求包采用本子包，其具体格式如下：

VerifySignedDataUpdateReq::= SEQUENCE {
signMethod　　INTEGER，
hashValueLen　　INTEGER，
hashValue　　OCTET STRING，
inDataLen　　INTEGER，
inData　　OCTET STRING
}

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

inDataLen 表示数据明文长度；

inData 表示数据明文。

——VerifySignedMessageUpdateResp 包：

VerifySignedMessageUpdateResp 包为多包验证消息签名更新响应格式包，当 respType 取值 verifySignedMessageUpdate 时，响应包采用本子包，其具体格式如下：

VerifySignedMessageUpdateResp::= SEQUENCE {
respValue　　INTEGER，
hashValue　　OCTET STRING OPTIONAL
}

respValue 表明响应码，0 表示成功，非 0 表示错误；

hashValue 表示杂凑值。

A.5.19 多包验证消息签名结束

——VerifySignedMessageFinalReq 包：

VerifySignedMessageReq 包为多包验证消息签名结束请求格式包，当 reqType 取值 verifySignedMessageFinal 时，请求包采用本子包，其具体格式如下：

```
VerifySignedMessageFinalReq::= SEQUENCE {
signMethod              INTEGER,
hashValueLen            INTEGER,
hashValue               OCTET STRING,
signature               OCTET STRING
}
```

signMethod 表明使用的签名算法类型，详细定义见 GM/T 0006—2012 中 6.2.4 签名算法标识；

type 表示使用验证消息签名时使用证书或证书序列号，1 表示使用证书，2 表示使用证书序列号；

cert 表示签名证书，type 取值 1 时有效；

certSN 表示签名证书序列号，type 取值 2 时有效；

hashValueLen 表示杂凑中间值长度；

hashValue 表示杂凑中间值；

signedMessage 表示消息签名数据，当公钥算法为 RSA 时，消息的结构遵循 PKCS#7；当公钥算法为 SM2 时，消息的结构遵循 GM/T 0010。

——VerifySignedMessageFinalResp 包：

VerifySignedMessageFinalResp 包为多包验证消息签名结束响应格式包，当 respType 取值 verifySignedMessageFinal 时，响应包采用本子包，其具体格式如下：

```
VerifySignedMessageFinalResp::= SEQUENCE {
respValue               INTEGER
}
```

respValue 表明响应码，0 表示成功，非 0 表示错误。

附 录 B
（规范性附录）
基于 HTTP 的签名消息协议语法规范

B.1 概述

附录 A 中描述的 ASN.1 格式是一种二进制格式，考虑到签名验证服务将被广泛用于各种 WEB 系统，而 WEB 系统更善于处理文本，为此在附录 A 的基础上，另行设计了一套基于 HTTP 协议消息协议接口，便于各类 WEB 系统调用。

其工作原理与附录 A 中的请求响应模式类似，不同的是将消息格式从二进制的 ASN.1 格式，转换为易于在 WEB 应用和 HTTP 协议中传递的文本格式。

本附录只描述了从附录 A 的消息格式到对应 HTTP 格式的转换规则，而不再复述附录 A 中每个请求，响应的业务含义。

B.2 ASN.1 数据类型到 HTTP 格式的转化规则

表 B.1 ASN.1 数据类型到 HTTP 格式的转化规则

ASN.1 类型	HTTP 字段类型	示 例
INTEGER	数字的 10 进制文本表示	比如 1 将被转换为"1"
BOOLEAN	文本 TRUE 和 FALSE	
GeneralizedTime	带时区格式的时间字符串 YYYYMMDDhhmm[ss(.s...)] {Z|+hhmm|-hhmm}	20131001120000Z+0800
OCTET STRING	对 OCTET STRING 进行 BASE64 编码后的结果	验证证书有效性申请包
Certificate	对 DER 格式的证书进行 BASE64 编码后的结果	

B.3 附录 A 中的请求与 HTTP 请求的转换规则

附录 A 中的请求分为两层结构，外层是公共结构。

```
SVSRequest ::= SEQUENCE {
version                 Version DEFAULT v1,
reqType                 ReqType,
request                 Request,
reqTime                 GeneralizedTime
}
```

这一层结构在转换为 HTTP 时，被转化为 HTTP Request Header 中的字段，原则如下：

a) 所有请求都采用 HTTP 的 POST 模式；

b) reqType 被作为 URL 的路径；

c) version 被作为一个自定义的 HTTP 字段 SVS-Request-Version；

d) reqTime 被作为一个自定义的 HTTP 字段 SVS-Request-Time。

一个转换的实例如下：

POST /SignDataHTTP/1.1\r\n

SVS-Request-Version：v1\r\n

SVS-Request-Time：20131001120000Z+0800\r\n

Content-Type：application/x-www-form-urlencoded\r\n

Content-Length：实际请求 body 长度\r\n

\r\n

signMethod=…

注 1：将 reqType 直接作为 WEB 路径，而不是自定义的 HTTP 字段，是为了能在通用的 WEB 负载设备上进行负载。

注 2：Content-Type 和 Content-Length 是 HTTP 协议的标准字段，此处按照其原意进行使用。

注 3：对应 HTTP 协议中的其他标准字段如 Host，User-Agent 等，与本附录无关，在此不再标出。

附录 A 中代表业务请求实体的 Request 类型，将被转换为一个 HTTP 的 form 表单：

param1=value1&param2=value2…paramN=valueN

比如 A.5.4 中的单包数字签名请求，将被转换为以下文本格式的表单：

signMethod=签名算法类型 &keyIndex=私钥的索引 &keyValue=私钥权限标识码 &inDataLen=1024&inData=Base64 编码的待签名的数据原文。

B.4 附录 A 中的响应与 HTTP 响应的转换规则

附录 A 中的响应分为两层结构，外层是公共结构。

```
SVSRespond ::= SEQUENCE {
version                 Version DEFAULT v1,
respType                RespType,
respond                 Respond,
respTime                GeneralizedTime
}
```

这一层结构在转换为 HTTP 时，被转化为 HTTP Response Header 中的字段，原则如下：

a) respType 被作为一个自定义的 HTTP 字段 SVS-Response-type；

b) version 被作为一个自定义的 HTTP 字段 SVS-Response-version；

c) respTime 被作为一个自定义的 HTTP 字段 SVS-Response-Time；

d) 代表业务响应实体的 Response 类型，也被转换为一个 HTTP 的 form 表单。

以附录 A 中 A.5.4 中的单包数字签名请求为例，其转换后 HTTP 响应如下：

HTTP 200 OK\r\n

SVS-Response-Type：SignData\r\n

SVS-Response-Version：v1\r\n

SVS-Response-Time：20131001120000Z+0800\r\n

Content-Type：text/html;charset=GB2312\r\n

Content-Length：实际响应 body 的长度\r\n

\r\n

respValue=0&signature=Base64 编码的签名结果。

附 录 C
（规范性附录）
响应码定义和说明

响应码定义和说明见表 C.1。

表 C.1 响应码定义和说明

宏 描 述	预定义值	说 明
GM_SUCCESS	0	正常返回
GM_ERROR_BASE	0x04000000	错误码起始值
GM_ERROR_CERT_ID	0x04000001	错误的证书标识
GM_ERROR_CERT_INFO_TYPE	0x04000002	错误的证书信息类型
GM_ERROR_SERVER_CONNECT	0x04000003	CRL 或 OCSP 服务器无法连接
GM_ERROR_SIGN_METHOD	0x04000004	签名算法类型错误
GM_ERROR_KEY_INDEX	0x04000005	签名者私钥索引值错误
GM_ERROR_KEY_VALUE	0x04000006	签名者私钥权限标识码错误
GM_ERROR_CERT	0x04000007	证书非法或服务器内不存在
GM_ERROR_CERT_DECODE	0x04000008	证书解码错误
GM_ERROR_CERT_INVALID_AF	0x04000009	证书过期
GM_ERROR_CERT_INVALID_BF	0x0400000A	证书尚未生效
GM_ERROR_CERT_REMOVED	0x0400000B	证书已被吊销
GM_INVALID_SIGNATURE	0x0400000C	签名无效
GM_INVALID_DATA_FORMAT	0x0400000D	数据格式错误
GM_SYSTEM_FAILURE	0x0400000E	系统内部错误
……	0x0400000F～0x040000FF	预留