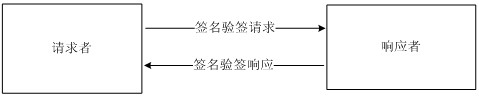
**1 范围**    本标准定义了签名验签服务器的相关术语，规定了签名验签服务器的功能要求、安全要求、接口要求、检测要求和消息协议语法规范等有关内容。   
    本标准适用于签名验签服务器的研制设计、应用开发和使用，也可用于指导签名验签服务器的检测。   
  
    **2 规范性引用文件**    下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件，凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。   
    GB/T 9813－2000 微型计算机通用规范   
    GB/T 19713-2005 信息技术 安全技术 公钥基础设施 在线证书状态协议   
    GM/T 0006 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 AAAAA PCI密码卡技术规范   
    GM/T BBBBB 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规范。   
  
    **3.1   
  
    签名验签服务器 Sign & Verify Server**    用于服务端的适用于公钥密码基础设施的服务器，其主要提供基于PKI体系和数字证书的数字签名、验证签名等运算功能，用于信息系统中，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。   
  
    **4 符号和缩略语**   
  
    下列符号和缩略语适用于本规范：   
    API 应用程序接口简称应用接口(Application Program Interface)   
    PKI 公钥密码基础设施   
    CA 证书认证机构   
    CRL 证书撤销列表   
    LDAP 轻量级目录访问协议   
    OCSP 在线证书状态查询协议   
    PKCS 公钥密码标准(Public-Key Cryptography Standard)   
  
    **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《信息技术 安全技术 公钥基础设施 在线证书状态协议》。   
  
    **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。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。   
  
   **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 《数字证书认证系统密码协议规范》。   
  
    **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。   
  
    **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使用的密码设备的管理界面实现。   
    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RSA算法时，数据的结构遵循PKCS#1标准或PKCS#7标准。   
    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 《SM2密码算法使用规范》或GM/T 0010 《SM2密码算法加密签名消息语法规范》。   
  
    **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认证机制，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认证。管理员登录成功后，通过管理界面进行初始化、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，以及认证失败、非法访问等异常事件的记录。   
  
   **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。   
  
    **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局审批的密码设备。调用密码设备的接口API应遵循GM/T 0018 《密码设备应用接口规范》。   
  
    **6.2 系统要求**   
  
    签名验签服务器所使用的操作系统，除满足服务器服务管理之外，其余功能应全部关闭或裁减。   
  
    **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，应通过身份认证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。管理员通过身份认证后，才能进行初始化、系统配置、应用管理、证书管理等操作。审计员通过身份认证后，才能进行日志审计等操作。   
  
    **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。   
  
    **6.7 应用编程接口API**   
  
    签名验签服务器的应用编程接口(API)必须遵循附录A《消息协议语法规范》或GM/T 0020 《证书应用综合服务接口规范》。应用实体可以通过TCP/IP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必须符合GB/T 9813－2000 《微型计算机通用规范》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20,000小时。   
  
    **7 签名验签服务器的检测要求**   
  
    检测要求规定了签名验签服务器的通用检测内容和方法。检测应包括外观和结构检查、提交文档的检查、功能检测、性能检验、敏感数据的保护与使用检测和物理检测等。   
  
    **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：   
    1) 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；   
    2) 开机自检的工作原理说明；   
    3) 自测程序的工作原理说明；   
    4) 敏感数据信息的存储和使用说明；   
    5) 物理防护措施说明；   
    6) 技术工作总结报告；   
    7) 安全性设计报告；   
    8) 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，检测结果符合本规范5.1和6.4.3.1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本规范5.2要求。   
  
    **7.3.3 应用管理功能检测**   
  
    签名验签服务器的用户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用户管理功能的检测结果应符合本规范5.3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本规范5.4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 《证书应用综合服务接口规范》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认证机制；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，检测结果应符合本规范5.6要求。   
  
    **7.3.7 日志管理检测**   
  
    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本规范5.7要求。   
  
   **7.3.8 系统自检检测**   
  
    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本规范5.8和6.4.3.2要求。   
  
    **7.3.9 应用编程接口（API）检测**   
  
    签名验签服务器的应用编程接口必须遵循附录A《消息协议语法规范》或GM/T 0020 《证书应用综合服务接口规范》。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。   
  
    **7.3.10 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，检测结果应符合本规范6.4.1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、1000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 《微型计算机通用规范》中“5.8环境试验”的要求进行，其结果应符合该规范中“6.8环境适应性”的要求。   
  
    **7.6 其它检测**   
  
    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**   
  
    本规范中，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**   
  
    **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**   
  
    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。   
  
    **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）,   
    verifySignedDataUpdat    （9）,   
    verifySignedDataFinal    （10）,   
    signMessage     （11）,   
    verifySignedMessagev     （12）,   
    signMessageInit      （13）,   
    signMessageUpdate     （14）,   
    signMessageFinal     （15）,   
    verifySignedMessageInit     （16）,   
    verifySignedMessageUpdate    （17）,   
    verifySignedMessageFinal    （18）   
    }   
    Request ::=CHOICE{   
    exportUserCertRe    [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，本节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1所示。

表1 请求包与请求类型值的对应关系

|  |  |  |
| --- | --- | --- |
| 请求类型字符描述 | 请求类型值 | 申请包说明 |
| 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 |
| 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,   
    respStatus            RespStatus ,   
    respType                      RespType,   
    respond                        Respond,   
    respTime              GeneralizedTime   
    }   
    其中：   
    Version ::= INTEGER { v1(0)}   
    RespStatus::= INTEGER{   
    okay         （0）,   
    error         （-1）,   
    }   
    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，本节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) 响应状态   
    响应状态给出响应者对请求者的处理结果，响应状态信息可用于人机交互。响应状态为0   
    表示响应成功，为-1表示响应失败。   
    c) 响应类型   
    本项描述了不同业务的响应类型值。   
    d) 响应包   
    响应包与响应类型值之间的对应关系如表2所示。

表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 |

   
    e) 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 《密码应用标识规范》中的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 《密码应用标识规范》中的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 《SM2密码算法使用规范》。   
  
    **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   
    }    
    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 《SM2密码算法使用规范》。   
    —— 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 《密码应用标识规范》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；   
    signerIDLen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 《密码应用标识规范》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 《密码应用标识规范》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 《SM2密码算法使用规范》。   
  
    **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 《密码应用标识规范》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；   
    signerIDLen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 《密码应用标识规范》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 I                 NTEGER,   
    type                                 INTEGER,   
    cert                                 [0] IMPLICT Certificate OPTIONAL,   
    certSN                           [1] IMPLICT OCTET STRING OPTIONAL,   
    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 《密码应用标识规范》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 《SM2密码算法使用规范》。   
    —— 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   
     }   
    signMethod表明使用的签名算法类型，详细定义见GM/T 0006 《密码应用标识规范》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。   
    —— 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 《SM2密码算法加密签名消息语法规范》。   
  
    **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   
    }   
    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 《SM2密码算法加密签名消息语法规范》。   
    —— 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 《密码应用标识规范》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；   
    signerIDLen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 《密码应用标识规范》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 《密码应用标识规范》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 《SM2密码算法加密签名消息语法规范》。   
  
    **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 《密码应用标识规范》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；   
    signerIDLen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 《密码应用标识规范》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 《密码应用标识规范》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 《SM2密码算法加密签名消息语法规范》。   
    —— 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**  
    响应码定义和说明 

|  |  |  |
| --- | --- | --- |
| 宏描述 | 预定义值 | 说明 |
| SVS\_SUCCESS | 0 | 正常返回 |
| SVS\_ERROR\_BASE | 0x04000000 |  |
| SVS\_ERROR\_CERT\_ID | 0x04000001 | 错误的证书标识 |
| SVS\_ERROR\_CERT\_INFO\_TYPE | 0x04000002 | 错误的证书信息类型 |
| SVS\_ERROR\_SERVER\_CONNECT | 0x04000003 | CRL或OCSP服务器无法连接 |
| SVS\_ERROR\_SIGN\_METHOD | 0x04000004 | 签名算法类型错误 |
| SVS\_ERROR\_KEY\_INDEX | 0x04000005 | 签名者私钥索引值错误 |
| SVS\_ERROR\_KEY\_VALUE | 0x04000006 | 签名者私钥权限标识码错误 |
| SVS\_ERROR\_CERT | 0x04000007 | 证书非法或服务器内不存在 |
| SVS\_ERROR\_CERT\_DECODE | 0x04000008 | 证书解码错误 |
| SVS\_ERROR\_CERT\_INVALID\_AF | 0x04000009 | 证书过期 |
| SVS\_ERROR\_CERT\_INVALID\_BF | 0x0400000A | 证书尚未生效 |
| SVS\_ERROR\_CERT\_REMOVED | 0x0400000B | 证书已被吊销 |
| SVS\_INVALID\_SIGNATURE | 0x0400000C | 签名无效 |
| SVS\_INVALID\_DATA\_FORMAT | 0x0400000D | 数据格式错误 |
| SVS\_SYSTEM\_FAILURE | 0x0400000E | 系统内部错误 |
| …… | 0x0400000F-  0x040000FF | 预留 |