ICS 35.030
CCS L 80

GM

中华人民共和国密码行业标准

GM/T 0136—2024

密码应用 HTTP 接口规范

HTTP interface specification for cryptographic application

2024-12-27 发布　　2025-07-01 实施

国家密码管理局　　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

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专利的责任。

本文件由密码行业标准化技术委员会提出并归口。

本文件起草单位：北京信安世纪科技股份有限公司、格尔软件股份有限公司、北京国脉信安科技有限公司、北京数字认证股份有限公司、中电科网络安全科技股份有限公司、浙江理工大学、武汉大学、北京格尔国信科技有限公司。

本文件主要起草人：焦靖伟、汪宗斌、郑强、袁峰、黄福飞、李虹霖、王银平、赵永省、罗俊、沈剑、胡进、罗敏、朱立通。

引　言

密码应用接口以 HTTP 形式对外提供，可以显著提升开发和使用的便利性。HTTP 接口简化了开发流程，开发者可以通过标准的 HTTP 请求直接与密码服务交互，降低编码复杂性。由于 HTTP 协议的广泛支持，接口实现了跨平台访问，能够在各种设备上灵活使用。HTTP 接口与多种编程语言和框架兼容，为开发者提供了灵活的工具选择。

密码应用 HTTP 接口在公钥密码基础设施支撑的前提下，向应用及系统提供各类通用的基础密码运算服务，有利于密码服务接口产品的开发，有利于应用系统在密码服务过程中的集成和实施，有利于各应用系统的互联互通。

密码应用 HTTP 接口规范

1 范围

本文件规定了通用密码应用支撑层密码应用对外 HTTP RESTful 接口定义、数据结构和接口描述。

本文件适用于多场景下密码应用对外提供 HTTP 服务，也适用于基础密码应用的开发和使用。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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 7408.1—2023 日期和时间 信息交换表示法 第 1 部分：基本原则

GB/T 17964—2021 信息安全技术 分组密码算法的工作模式

GB/T 25069—2022 信息安全技术 术语

GB/T 38636—2020 信息安全技术 传输层密码协议（TLCP）

GB/T 41389—2022 信息安全技术 SM9 密码算法使用规范

GM/T 0006—2023 密码应用标识规范

GM/T 0009—2023 SM2 密码算法使用规范

GM/T 0094—2020 公钥密码应用技术体系框架规范

GM/Z 4001—2013 密码术语

3 术语和定义

GB/T 25069 和 GM/Z 4001 界定的以及下列术语和定义适用于本文件。

3.1

密码应用 cryptographic application

信息系统中利用密码技术提供数据安全保护的应用程序或服务。

注：对外可提供基础密码运算接口，涵盖消息摘要、加解密、签名验签和证书解析等，确保信息的机密性、完整性、真实性和不可否认性。

4 缩略语

下列缩略语适用于本文件。

API：应用程序接口，简称应用接口（Application Program Interface）

HTTP：超文本传输协议（Hypertext Transfer Protocol）

JSON：JS 对象标记（JavaScript Object Notation）

NTV：名称类型值数据（Name-Type-Value Data）

RESTful：表现层状态转化原则（Representational State Transfer）

TLCP:传输层密码协议(Transport Layer Cryptography Protocol)

URL:统一资源定位符(Uniform Resource Locator)

5 接口通则

5.1 密码应用 HTTP 接口在公钥密码应用技术体系框架中的位置

公钥密码应用技术体系框架由应用层、典型密码应用支撑层、通用密码应用支撑层、基础设施安全支撑平台、密码设备服务层组成。密码应用 HTTP 接口属于通用密码应用支撑层中通用密码服务,向典型密码服务层和应用层提供证书解析、信息的机密性、完整性和不可否认性基础密码服务,将上层应用的密码服务请求转化为具体的基础密码操作请求,通过统一的密码设备应用接口调用相应的密码设备实现具体的密码运算,通用密码服务在公钥密码应用技术框架内的位置应遵循 GM/T 0094—2020 的第 4 章。

5.2 功能说明

密码应用支撑层密码应用通过 HTTP 接口,将外部应用的密码服务请求转化为具体的基础密码操作请求,通过统一的密码设备应用接口调用相应的密码设备实现具体的密码运算。

密码应用 HTTP 接口适用于各类密码产品对外提供 HTTP 形式的密码服务接口,涵盖签名验签、加解密、消息杂凑、消息鉴别和证书解析等基础密码服务功能。

接口遵循 RESTful 架构风格,基于 HTTP 协议,请求和响应数据采用 JSON 格式,使用多个 NTV 结构进行参数传递。

密码应用在设计时应确保接口遵循 RESTful 原则,具体包括:

a) 独立的资源标识:每个具体的服务接口都应视为独立的资源,并通过特定的 URL 路径进行标识;
b) 接口的统一性:使用标准的 HTTP 方法(GET、POST 等)对资源进行操作,不同的操作对应不同的方法,保证接口的一致性和规范性;
c) 接口的无状态性:服务器不保存客户端的上下文信息,每次请求都应包含完整的信息,使得密码应用服务提供方能够独立处理每个请求;
d) 统一的资源表示:采用 JSON 格式作为请求和响应数据的格式,通过多个 NTV 结构进行参数传递,确保数据的结构化和可读性,方便不同系统之间的数据交互和解析。

6 接口定义

6.1 通信协议

密码应用 HTTP 接口基于 HTTP 协议,为保证传输安全性,应采用 TLCP 等协议进行传输层安全保护。如使用 TLCP,应符合 GB/T 38636。

6.2 URL 规则和请求方法

所有请求的 URL 路径应以“/shf/v1/”为前缀,后跟具体的资源路径。shf 表示密码应用 HTTP 接口,当前版本为 v1。随着本文件的修订,版本号也应相应增加。除本文件约定的接口外,自定义扩展接口也应按照本文件的格式要求实现。例如,自定义扩展接口的请求地址前缀可以为“/shf/ext/”。

请求地址、请求体和响应参数大小写敏感。

应支持 HTTP GET、POST 方法。

a) 使用 GET 方法时，无需添加请求参数，响应体应为 JSON 格式。示例 1 给出了 GET 方法的 HTTP 请求体。

示例 1：

```
GET /shf/v1/GetAsymmetricInfo HTTP/1.1
```

b) 使用 POST 方法时，请求体和响应体都应采用 JSON 格式。示例 2 给出了 POST 方法的 HTTP 请求体。

示例 2：

```
POST /shf/v1/Encrypt HTTP/1.1
Content-Type: application/json
{
  "Data": [
    {
      "Name": "CipherText",
      "Type": "Hex",
      "Value": "307a022100a92d9ec36b4db84eecf0b9561e875d2dd69cfaa26bebb0cae41bb92fc40baa6d022100fbcd
6be7755352c96b60beb8a183b773e3faa50bee20bf6ded1086ff069777b004207f7d2bbd48bfc1d4e8b41f0a8830c94dc3e76
f4c4ca7b28defb84693a3fdc0b004108417f00191a723e27b6dd1d9986bba20"
    },
    {
      "Name": "KeyType",
      "Type": "Raw",
      "Value": "SGD_SM4"
    }
  ]
}
```

6.3 数据结构

6.3.1 请求数据结构

请求体应包含请求数据，默认为 UTF-8 编码。数据(Data)中应传递请求参数，NTV 结构应以数组形式存于 Data 中。请求应通过多个 NTV 结构进行参数的传递。NTV 结构应符合附录 A 的规定，示例给出了请求数据结构的 JSON 实例。

示例：

```
{
  "Data": [
    {
      "Name": "name",
      "Type": "Hex",
      "Value": "76616c756531"
    }
  ]
}
```

6.3.2 响应数据结构

响应体内容应包含状态和数据两部分，分别以 Status 和 Data 定义。状态(Status)中应存储状态代码和状态消息，状态代码(Code)应为十六进制编码字符串，其定义见附录 B。状态消息(Msg)应为可读字符串，数据(Data)中应传递服务端响应数据，NTV 结构应以数组形式存于 Data 中。示例给出了相应数据结构的 JSON 实例。

当请求中未指定响应的字符编码格式时，应采用 UTF-8 编码。

当没有数据需要响应时，Data 字段应为空。

示例：

```
{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    // NTV 响应数据
  ]
}
```

7 接口描述

7.1 接口列表

本文件采用 HTTP RESTful API 风格描述。

HTTP 响应状态码描述了请求的结果和响应的语义。状态代码描述业务具体错误类型，其格式和内容应符合附录 B 的规定。接口请求及响应相关样例见附录 C。

密码应用 HTTP 接口 API 见表 1。

表 1 密码应用 HTTP 接口 API

序号	接口功能	请求路径	方法
1	获取非对称算法列表	/shf/v1/GetAsymmetricInfo	GET
2	外部公钥加密	/shf/v1/EncryptByExternalPublicKey	POST
3	内部公钥加密	/shf/v1/EncryptByInternalPublicKey	POST
4	内部私钥签名	/shf/v1/Sign	POST
5	外部公钥验签	/shf/v1/VerifyByExternalPublicKey	POST
6	内部公钥验签	/shf/v1/VerifyByInternalPublicKey	POST
7	获取对称算法列表	/shf/v1/GetSymmetricInfo	GET
8	对称密钥加密	/shf/v1/Encrypt	POST
9	对称密钥解密	/shf/v1/Decrypt	POST
10	获取杂凑算法列表	/shf/v1/GetHashInfo	GET
11	消息杂凑	/shf/v1/Hash	POST

表 1　密码应用 HTTP 接口 API（续）

序号	接口功能	请求路径	方法
12	产生随机数	/shf/v1/RetrieveRandom	POST
13	获取消息鉴别算法列表	/shf/v1/GetMacInfo	GET
14	基于杂凑算法的消息鉴别	/shf/v1/HMac	POST
15	基于分组密码的消息鉴别	/shf/v1/CipherMac	POST
16	获取证书解析标签列表	/shf/v1/GetCertInfo	GET
17	证书解析	/shf/v1/ParseCertificate	POST

部分 HTTP 接口支持大数据量的密码运算，请求参数同时支持消息和文件地址，其中：

——服务端不支持直接上传文件，而是以文件地址作为请求参数，应确保服务端可正常访问文件地址；

——请求参数应为消息或文件地址中的一种，两种不可同时出现；

——双方应确保请求响应过程中任何文件不被第三方获取。

加解密相关接口（外部公钥加密、内部公钥加密、对称密钥加密、对称密钥解密）消息和文件地址的请求响应对应关系为：

——加密消息请求参数为 PlainText 时，响应参数应采用 CipherText；

——加密文件请求参数为 FileUri 时，响应参数应采用 CallbackUri；

——解密消息请求参数为 CipherText 时，响应参数应采用 PlainText；

——解密文件请求参数为 FileUri 时，响应参数应采用 CallbackUri。

7.2　密码运算类接口

7.2.1　获取非对称算法列表

获取非对称算法列表接口定义应符合表 2 的规定。

表 2　获取非对称算法列表接口定义

请求路径	/shf/v1/GetAsymmetricInfo		
调用方法	GET		
功能描述	获取服务端支持的非对称算法列表，包含密钥类型，可包含密钥用法		
请求参数	无		
—	参数	类型	说明
响应参数	KeyTypes	Raw	密钥类型数组，应符合 GM/T 0006—2023 的 6.2.2 中标签的定义
	KeyUsages	Raw	可选，密钥用法数组，例如：签名、加密等。 加密密钥用法应符合 GM/T 0006—2023 的 6.2.2 中标签的定义； 签名密钥用法应符合 GM/T 0006—2023 的 6.2.4 中标签的定义

7.2.2　外部公钥加密

外部公钥加密接口定义应符合表 3 的规定。

表 3　外部公钥加密接口定义

请求路径	/shf/v1/EncryptByExternalPublicKey		
调用方法	POST		
功能描述	使用外部公钥加密消息或文件		
—	参数	类型	说明
请求参数	PublicKey	Base64/Hex	公钥信息。公钥封装格式 SM2 应符合 GM/T 0009，SM9 应符合 GB/T 41389
	PlainText	Raw/Base64/Hex	待加密明文，消息
	FileUri	Raw	待加密明文，文件地址字符串
响应参数	CipherText	Base64/Hex	明文消息的密文
	CallbackUri	Raw	加密文件回调地址字符串

7.2.3　内部公钥加密

内部公钥加密接口定义应符合表 4 的规定。

表 4　内部公钥加密接口定义

请求路径	/shf/v1/EncryptByInternalPublicKey		
调用方法	POST		
功能描述	使用内部公钥加密消息或文件		
—	参数	类型	说明
请求参数	KeyId	Raw	密钥 Id
	PlainText	Raw/Base64/Hex	待加密明文，消息
	FileUri	Raw	待加密明文，文件地址字符串
响应参数	CipherText	Base64/Hex	明文消息的密文
	CallbackUri	Raw	加密文件回调地址字符串

7.2.4　内部私钥签名

内部私钥签名接口定义应符合表 5 的规定。

表 5　内部私钥签名接口定义

请求路径	/shf/v1/Sign		
调用方法	POST		
功能描述	使用内部私钥签名消息或文件		
—	参数	类型	说明
请求参数	KeyId	Raw	私钥 Id
	PlainText	Raw/Base64/Hex	待签名明文，消息
	FileUri	Raw	待签名明文，文件地址字符串
响应参数	Signature	Base64/Hex	签名值

7.2.5 外部公钥验签

外部公钥验签接口定义应符合表 6 的规定。

表 6 外部公钥验签接口定义

请求路径	/shf/v1/VerifyByExternalPublicKey		
调用方法	POST		
功能描述	使用外部公钥验签		
—	参数	类型	说明
请求参数	Algorithm	Raw	签名算法标识,应符合 GM/T 0006—2023 标签的定义
	PlainText	Raw/Base64/Hex	待验证明文,消息
	PublicKey	Base64/Hex	公钥信息。公钥封装格式 SM2 应符合 GM/T 0009,SM9 应符合 GB/T 41389
	FileUri	Raw	待验证明文,文件地址字符串
	Signature	Base64/Hex	签名值
响应参数	SignatureValid	Raw	验签结果,应为布尔值

7.2.6 内部公钥验签

内部公钥验签接口定义应符合表 7 的规定。

表 7 内部公钥验签接口定义

请求路径	/shf/v1/VerifyByInternalPublicKey		
调用方法	POST		
功能描述	使用内部公钥验签		
—	参数	类型	说明
请求参数	KeyId	Raw	密钥 Id
	PlainText	Raw/Base64/Hex	待验证明文,消息
	FileUri	Raw	待验证明文,文件地址字符串
	Signature	Base64/Hex	签名值。签名值格式 SM2 应符合 GM/T 0009,SM9 应符合 GB/T 41389
响应参数	SignatureValid	Raw	验签结果,应为布尔值

7.2.7 获取对称算法列表

获取对称算法列表接口定义应符合表 8 的规定。

表 8　获取对称算法列表接口定义

请求路径	/shf/v1/GetSymmetricInfo		
调用方法	GET		
功能描述	获取服务端支持的对称算法列表，包含密钥类型和长度、分组算法标识和填充方法等		
请求参数	无		
—	参数	类型	说明
响应参数	KeyTypes	Raw	支持的密钥类型(含密钥长度)数组。 密钥类型格式：××××_×××(密钥类型_密钥长度)，密钥类型数组，应符合 GM/T 0006—2023 的 6.2.1 中标签的定义
	Algorithms	Raw	分组密码算法标识数组，应符合 GM/T 0006 6.2.1 中标签的定义
	Paddings	Raw	填充方法，见 GB/T 17964—2021 的附录 C

7.2.8　对称密钥加密

对称密钥加密接口定义应符合表 9 的规定。

表 9　对称密钥加密接口定义

请求路径	/shf/v1/Encrypt		
调用方法	POST		
功能描述	使用内部对称密钥加密明文消息或文件。此接口可用于可鉴别加密的 CCM 和 GCM 模式，相关请求和响应参数应按照 GB/T 36624 的规定进行确定		
—	参数	类型	说明
请求参数	KeyId	Raw	对称密钥 Id
	Algorithm	Raw	分组密码算法标识，应符合 GM/T 0006—20234 的 6.2.1 中标签的定义
	PlainText	Raw/Base64/Hex	待加密明文，消息
	FileUri	Raw	待加密明文，文件地址字符串
	Padding	Raw	可选，填充方法
	IV	Raw/Base64/Hex	可选，初始向量
	AAD	Raw/Base64/Hex	可选，用于可鉴别加密，额外的可鉴别数据
	Tag	Raw	可选，用于可鉴别加密 GCM 模式，GCM Tag 比特长度，长度应遵循 GB/T 36624
响应参数	CipherText	Base64/Hex	消息密文
	CallbackUri	Raw	加密后文件回调地址字符串

7.2.9　对称密钥解密

对称密钥解密接口定义应符合表 10 的规定。

表 10　对称密钥解密接口定义

请求路径	/shf/v1/Decrypt		
调用方法	POST		
功能描述	使用内部对称密钥解密消息密文或加密文件。此接口可用于可鉴别加密的 CCM 和 GCM 模式，相关请求和响应参数应按照 GB/T 36624 的规定进行确定		
—	参数	类型	说明
请求参数	KeyId	Raw	对称密钥 Id
	Algorithm	Raw	分组密码算法标识，应符合 GM/T 0006—2023 的 6.2.1 中标签的定义
	CipherText	Base64/Hex	待解密密文，消息
	FileUri	Raw	待解密密文，文件地址字符串
	Padding	Raw	可选，填充方法
	IV	Raw/Base64/Hex	可选，初始向量
	AAD	Raw/Base64/Hex	可选，用于可鉴别加密，额外的可鉴别数据
	Tag	Raw	可选，用于可鉴别加密 GCM 模式，GCM Tag 比特长度，长度应遵循 GB/T 36624
响应参数	PlainText	Raw/Base64/Hex	消息明文
	CallbackUri	Raw	解密后文件回调地址字符串

7.2.10　获取杂凑算法列表

获取杂凑算法列表接口定义应符合表 11 的规定。

表 11　获取杂凑算法列表接口定义

请求路径	/shf/v1/GetHashInfo		
调用方法	GET		
功能描述	获取服务端支持的杂凑算法列表，包含杂凑算法标识		
请求参数	无		
—	参数	类型	说明
响应参数	HashAlgorithms	Raw	支持的杂凑算法标识数组，应符合 GM/T 0006—2023 的 6.2.3 中标签的定义

7.2.11　消息杂凑

消息杂凑接口定义应符合表 12 的规定。

表 12　消息杂凑接口定义

请求路径	/shf/v1/Hash		
调用方法	POST		
功能描述	对给定消息或者文件杂凑		
—	参数	类型	说明
请求参数	Algorithm	Raw	杂凑算法标识
	Message	Raw/Base64/Hex	待杂凑消息
	FileUri	Raw	待杂凑文件地址字符串
响应参数	Digest	Base64/Hex	杂凑值

7.2.12　产生随机数

产生随机数接口定义应符合表 13 的规定。

表 13　产生随机数接口定义

请求路径	/shf/v1/RetrieveRandom		
调用方法	POST		
功能描述	获取指定长度随机数		
—	参数	类型	说明
请求参数	Length	Raw	随机数长度，应为整数
响应参数	Random	Base64/Hex	随机数

7.2.13　获取消息鉴别算法列表

获取消息鉴别算法列表接口定义应符合表 14 的规定。

表 14　获取消息鉴别算法列表接口定义

请求路径	/shf/v1/GetMacInfo		
调用方法	GET		
功能描述	获取服务端支持的消息鉴别算法列表，包含基于杂凑算法的消息鉴别标识和基于分组密码的消息鉴别标识		
请求参数	无		
—	参数	类型	说明
响应参数	HMacAlgorithms	Raw	支持的基于杂凑算法的消息鉴别标识，应符合 GM/T 0006—2023 的 6.2.3 中标签的定义
	CipherMacAlgorithms	Raw	支持的基于分组密码的消息鉴别标识，应符合 GM/T 0006—2023 的 6.2.1 中标签的定义

7.2.14 基于杂凑算法的消息鉴别

基于杂凑算法的消息鉴别接口定义应符合表 15 的规定。

表 15 基于杂凑算法的消息鉴别接口定义

请求路径	/shf/v1/HMac		
调用方法	POST		
功能描述	基于杂凑算法的消息鉴别		
—	参数	类型	说明
请求参数	KeyId	Raw	密钥 Id
	Algorithm	Raw	杂凑算法的消息鉴别标识
	Message	Raw/Base64/Hex	待认证消息
响应参数	Mac	Base64/Hex	消息鉴别码

7.2.15 基于分组密码的消息鉴别

基于分组密码的消息鉴别接口定义应符合表 16 的规定。

表 16 基于分组密码的消息鉴别接口定义

请求路径	/shf/v1/CipherMac		
调用方法	POST		
功能描述	基于分组密码的消息鉴别		
—	参数	类型	说明
请求参数	KeyId	Raw	密钥 Id
	Algorithm	Raw	基于分组密码的消息鉴别标识
	Message	Raw/Base64/Hex	待认证消息
	Nonce	Raw/Base64/Hex	可选，随机数
	AAD	Raw/Base64/Hex	可选，额外的可鉴别数据
响应参数	Mac	Base64/Hex	消息鉴别码

7.3 证书解析类接口

7.3.1 获取证书解析标签列表

获取证书解析标签列表接口定义应符合表 17 的规定。

表 17　获取证书解析标签列表接口定义

请求路径	/shf/v1/GetCertInfo		
调用方法	GET		
功能描述	获取服务端支持的证书解析标签列表		
请求参数	无		
—	参数	类型	说明
响应参数	Tags	Raw	支持的证书解析项标签数组，应符合 GM/T 0006—2023 的 6.3.4 中标签的定义

7.3.2　证书解析

证书解析接口定义应符合表 18 的规定。

表 18　证书解析接口定义

请求路径	/shf/v1/ParseCertificate		
调用方法	POST		
功能描述	获取证书指定标签的内容		
—	参数	类型	说明
请求参数	Certificate	Raw/Base64/Hex	待解析证书
	Tags	Raw	证书解析项标签，该项为空则解析所有标签。应符合 GM/T 0006—2023 中标签的定义
响应参数	—	Raw	请求参数中的解析项标签，不存在时不响应该标签内容

附 录 A
（规范性）
NTV 结构

A.1 概述

密码运算中常见各种二进制数据，使用 JSON 数据交换格式传输时，需要将各种二进制数据编码为文本格式。编码方式常见的有 Base64 编码和十六进制编码等。为保证兼容性和灵活性，不宜限制数据的编码类型。因此，采用名称，类型，值的方式表示一个最小数据单元。以 Golang 语言为例，NTV 结构表示为：

```
type NTV struct {
    Name        string          `json:"Name"`
    Type        string          `json:"Type"`
    Value       interface{}     `json:"Value"`
}
```

每个请求由多个 NTV 结构组成，每个 NTV 结构包含三项，依次为参数名称、参数类型和参数值。各参数定义如下。

A.2 参数名称

参数名称是由服务端定义的字符串，用于标识请求参数。

A.3 参数类型

将二进制数据转化为可读字符串传递，需要对二进制数据进行编码。如未指定参数类型，则应使用默认 Raw 参数类型。当参数类型为 Raw 时，可以省略传输该参数类型的字段。

允许的参数类型值见表 A.1。

表 A.1 允许的参数类型值

参数类型	参数值样例	说明
Raw	"test"	参数值数据类型应为字符串，数字、对象、数组、布尔值的一种。不可为空
	["SM3","SHA256"]	
	true/false	
	123456789	
Base64	dGVzdA==	—
Hex	74657374	—
DataTime	2001-01-01T10:00:00+10:00	可选

A.4 参数值

参数值是由参数类型格式或者按照参数类型格式编码的字符串，具体取决于参数类型。参数类型和参数值应符合：

——Raw，可读字符串或者常见 JSON 原语允许格式。示例 1 给出了 Raw 类型的 NTV 结构

JSON 实例。

示例 1：

```
{"Name":"HashAlgorithms","Type":"Raw","Value":["SM3","SHA256","SHA512"]}
```

——Base64，二进制数据经 Base64 编码后的字符串。示例 2 给出了 Base64 类型的 NTV 结构 JSON 实例。

示例 2：

```
{"Name": "PlainText", "Type":"Base64", "Value":"dHJhbnNmZXI="}
```

——Hex，二进制数据经十六进制编码后的字符串。示例 3 给出了 Hex 类型的 NTV 结构 JSON 实例。

示例 3：

```
{"Name": "Plaintext", "Type":"Hex", "Value":"76616c756531"}
```

——DateTime，符合 GB/T 7408.1 日期时间字符串。忽略或者截断小数秒，需包含时区。示例 4 给出了 DateTime 类型的 NTV 结构 JSON 实例。

示例 4：

```
{"Name": "Time", "Type":"DateTime", "Value":"2001-01-01T10:00:00+10:00"}
```

附 录 B
（规范性）
状态代码

状态代码定义见表 B.1。

表 B.1 状态代码定义

状态代码	说明
0	成功
0x0E000000	未知错误
0x0E000001	客户端错误
0x0E000002	服务端错误
0x0E000003	通信异常
0x0E000004	接口认证失败
0x0E000005	请求参数错误
0x0E000006	NTV 解析失败
0x0E000007	密钥权限异常
0x0E000008	密钥类型错误
0x0E000009	不存在的密钥调用
0x0E00000A	不支持的算法
0x0E00000B	公钥运算失败
0x0E00000C	私钥运算失败
0x0E00000D	非对称加密数据错误
0x0E00000E	签名运算失败
0x0E00000F	验证签名失败
0x0E000010	消息杂凑运算失败
0x0E000011	随机数产生失败
0x0E000012	HMac 运算失败
0x0E000013	CipherMac 运算失败
0x0E000014	对称算法运算失败
0x0E000015	证书解析失败
0x0E000016	请求数据超出限制
0x0E000017	文件长度超出限制
0x0E000018	指定的文件不存在
0x0E000019	客户端权限不足
0x0E00001A	请求的资源不存在
0x0E00001B	资源已删除

表 B.1　状态代码定义（续）

状态代码	说明
0x0E00001C	请求频率超限
0x0E00001D	并发请求数超限
0x0E00001E	请求配额已用尽
0x0E00001F	HTTP 请求方法错误
0x0E000020～0x0EFFFFFF	预留

附 录 C
（资料性）
接口样例

C.1 密码运算类接口样例

C.1.1 获取非对称算法列表


```
{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "KeyTypes",
      "Type": "Raw",
      "Value": [
        "SGD_SM2",
        "SGD_SM9"
      ]
    },
    {
      "Name": "KeyUsages",
      "Type": "Raw",
      "Value": [
        "Encrypt",
        "Sign"
      ]
    }
  ]
}
```


C.1.2 外部公钥加密


```
// 文本
{
  "Data": [
    {
      "Name": "PublicKey",
      "Type": "Base64",
      "Value": "A0IABD2An7SH2UJQZ3xWlYrq3uho796PaWHneSuXxy6zLx7oJq5Gzzc83vNN/
Q8KywdNouWWSJj8Hg2klBEh1q5vxNU="
```

```
      },
      {
        "Name": "PlainText",
        "Type": "Raw",
        "Value": "Hello World"
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "CipherText",
        "Type": "Base64",
        "Value": "MHUCIQCoTWfkQcfsrKqbqwI96GdYFcaxKgYikka3lhbVmM/F8AIhANTQej
O4MdcachZVKV+wSoxKOrCI/MLQp6ViPdnX2MRRBCCg0odV/LSkzDJfYCMpaqcRqpY4V2bOsIr
uTNPBltVgxQQLtPaHzGmK6VvTErg="
      }
    ]
  }

// 文件
  {
    "Data": [
      {
        "Name": "PublicKey",
        "Type": "Base64",
        "Value": "A0IABBGIJwA4GitubP4klUnOC+Ug73/q2ZW2TuVBLR/+Qq7LmsWECwRJ
EjeCbs/CFZiMD1W9aVKwiy0oELPfZKcHY/s="
      },
      {
        "Name": "FileUri",
        "Type": "Raw",
        "Value": "file:///Users/Gsealy/Desktop/test.txt"
      }
    ]
  }

  {
```

```
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "CallbackUri",
      "Type": "Raw",
      "Value": "https://localhost:8000/test.txt.encrypted"
    }
  ]
}
```


C.1.3 内部公钥加密


```
{
  "Data": [
    {
      "Name": "KeyId",
      "Type": "Raw",
      "Value": "e772fb99-712b-4e9a-8923-900b6e8715bf"
    },
    {
      "Name": "PlainText",
      "Type": "Raw",
      "Value": "Hello World"
    }
  ]
}

{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "CipherText",
      "Type": "Base64",
      "Value": "MHQCIQDUofbj2/ctLYZ5DMgjVk3B6nojaTDX7GkKiDEPi35LmgIgU0Irv3whqGvb72xUnaJF/iZors7uFH7f8ql0WlAGZHAEIE3Ga74Jl2l5zeCNCtri4uru6vA0wrnftQ4YrCxug1dgBAt4aZ21oskQvg9HAg=="
    }
  ]
```

```
}
```


C.1.4 内部私钥签名


```
{
  "Data": [
    {
      "Name": "KeyId",
      "Type": "Raw",
      "Value": "9bcf0c91-f9f1-406d-ab69-d7bbc157cc06"
    },
    {
      "Name": "PlainText",
      "Type": "Raw",
      "Value": "Hello World"
    }
  ]
}

{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "Signature",
      "Type": "Base64",
      "Value": "MEQCID0G9Qh9tXfhqOfv4kXuIZvm45U+Y7BFbufFZDNJvJHZAiBgkdtAxzrB
3J5nJD3wmiF0yVzudEt6cYl6ZLXE//4dSQ=="
    }
  ]
}
```


C.1.5 外部公钥验签


```
{
  "Data": [
    {
      "Name": "PublicKey",
      "Type": "Base64",
      "Value": "A0IABA+llOsgG0nwUtjz2U7gUOCD2KUq+Vs+eTkqWA/fRviKN/R2KCf5p
owOzrlQvCh9+wQeRAODX5jNE+zFWjUQwuE="
    },
    {
```

```
        "Name": "PlainText",
        "Type": " Base64",
        "Value": "SGVsbG8gV29ybGQ="
      },
      {
        "Name": "Signature",
        "Type": "Base64",
        "Value": "MEUCIQD4djuzX1cxYcZtn6x+yh90xc+F6hLdiTTmBD34BkFHIAIgHZIcZvkd
dzqshHl4hdK63nl209I299fq9bKhjMAQDyg="
      },
      {
        "Name": "Algorithm",
        "Type": "Raw",
        "Value": "1.2.156.10197.1.501"
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "SignatureValid",
        "Type": "Raw",
        "Value": true
      }
    ]
  }
```


C.1.6 内部公钥验签


```
  {
    "Data": [
      {
        "Name": "KeyId",
        "Type": "Raw",
        "Value": "9bcf0c91-f9f1-406d-ab69-d7bbc157cc06"
      },
      {
        "Name": "PlainText",
        "Type": "Raw",
```

```
        "Value": "Hello World"
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Signature",
        "Type": "Base64",
        "Value": "MEQCID0G9Qh9tXfhqOfv4kXuIZvm45U+Y7BFbufFZDNJvJHZAiBgkdtAxzr
B3J5nJD3wmiF0yVzudEt6cYl6ZLXE//4dSQ==="
      }
    ]
  }
```


C.1.7 获取对称算法列表


```
  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "KeyTypes",
        "Type": "Raw",
        "Value": [
          "SGD_SM1",
          "SGD_SM4"
        ]
      },
      {
        "Name": "Algorithms",
        "Type": "Raw",
        "Value": [
          "SGD_SM1_ECB",
          "SGD_SM4_CBC",
          "SGD_SM4_CFB",
          "SGD_SM4_CTR"
```

```
        ]
      },
      {
        "Name": "Paddings",
        "Type": "Raw",
        "Value": [
          "ZERO",
          "NONE"
        ]
      }
    ]
  }
```


C.1.8 对称密钥加密


```
  {
    "Data": [
      {
        "Name": "Algorithm",
        "Type": "Raw",
        "Value": "SGD_SM4_CBC"

      },
      {
        "Name": "PlainText",
        "Type": "Base64",
        "Value": "aGVsbG8gd29ybGQ="
      },
      {
        "Name": "KeyId",
        "Type": "Raw",
        "Value": "d043ee15-747c-4f4b-92a6-2464a171bad3"
      },
      {
        "Name": "Padding",
        "Type": "Raw",
        "Value": "ZERO"
      },
      {
        "Name": "IV",
        "Type": "Base64",
        "Value": "7MEQBcZd/t6kbQ42UyYRqw=="
      }
    ]
```

```
}

{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "CipherText",
      "Type": "Hex",
      "Value": "15fdcc8f75bc6af137b18ce2cc85bfb7"
    }
  ]
}
```

C.1.9 对称密钥解密

```
{
  "Data": [
    {
      "Name": "Algorithm",
      "Type": "Raw",
      "Value": "SGD_SM4_CBC"
    },
    {
      "Name": "CipherText",
      "Type": "Base64",
      "Value": "1nhzbqij5V/s9wq3q9TVIA==="
    },
    {
      "Name": "KeyId",
      "Type": "Raw",
      "Value": "f4bf8f9d-3755-4b08-b6f0-dd75a4f75cf5"
    },
    {
      "Name": "Padding",
      "Type": "Raw",
      "Value": "ZERO"
    },
    {
      "Name": "IV",
      "Type": "Base64",
      "Value": "OTT7HUpsYLGvuBWG6HPIJQ=="
```

```
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "PlainText",
        "Type": "Hex",
        "Value": "68656c6c6f20776f726c64"
      }
    ]
  }
```

C.1.10 获取杂凑算法列表

```
  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "HashAlgorithms",
        "Type": "Raw",
        "Value": [
          "SGD_SM3",
          "SGD_SHA256",
          "SGD_SHA384"
        ]
      }
    ]
  }
```

C.1.11 消息杂凑

```
  {
    "Data": [
      {
        "Name": "Algorithm",
        "Type": "Raw",
```

```
        "Value": "SGD_SM3"
      },
      {
        "Name": "Message",
        "Type": "Raw",
        "Value": "hello world"
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Digest",
        "Type": "Hex",
        "Value": "44f0061e69fa6fdfc290c494654a05dc0c053da7e5c52b84ef93a9d67d3fff88"
      }
    ]
  }
```


C.1.12 产生随机数


```
  {
    "Data": [
      {
        "Name": "Length",
        "Type": "Raw",
        "Value": 256
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Random",
```

```
        "Type": "Base64",
        "Value": "NY+PHERe+Wh4tq+ZApLcGILcL6+yQsRHtUHy1XKSaIA="
      }
    ]
  }
```


C.1.13 获取消息鉴别算法列表


```
  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "HMacAlgorithms",
        "Type": "Raw",
        "Value": [
          "SGD_SM3_HMAC",
          "SGD_SHA256_HMAC"
        ]
      },
      {
        "Name": "CipherMacAlgorithms",
        "Type": "Raw",
        "Value": [
          "SGD_SM4_GCM",
          "SGD_SM4_CCM"
        ]
      }
    ]
  }
```


C.1.14 基于杂凑算法的消息鉴别


```
  {
    "Data": [
      {
        "Name": "Algorithm",
        "Type": "Raw",
        "Value": "SGD_SM3_HMAC"
      },
      {
        "Name": "Message",
        "Type": "Base64",
```

```
        "Value": "aGVsbG8gd29ybGQ="
      },
      {
        "Name": "KeyId",
        "Type": "Raw",
        "Value": "3d3bfa99-4eee-4422-a6f3-e8cf4e323595"
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Mac",
        "Type": "Base64",
        "Value": "Fl4+n2hPfbE+KLpG+1e48vqCNCvAlJXCDeaIL+8J9SU="
      }
    ]
  }
```


C.1.15 基于分组密码的消息鉴别


```
  {
    "Data": [
      {
        "Name": "Algorithm",
        "Type": "Raw",
        "Value": "SGD_SM4_GCM"
      },
      {
        "Name": "Message",
        "Type": "Base64",
        "Value": "aGVsbG8gd29ybGQ="
      },
      {
        "Name": "KeyId",
        "Type": "Raw",
        "Value": "071dca46-df98-4429-bb52-54922c77bb8f"
      },
      {
```

```
        "Name": "Nonce",
        "Type": "Base64",
        "Value": "3TCLEvzBzpgft+xAoOc+Xg=="
      }
    ]
  }

  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Mac",
        "Type": "Base64",
        "Value": "BkMxrUxlhJIx/WbjHwv4s4VvOQogCKROVtC2"
      }
    ]
  }
```


C.2 证书解析类接口

C.2.1 获取证书解析标签列表


```
  {
    "Status": {
      "Code": "0",
      "Msg": "success"
    },
    "Data": [
      {
        "Name": "Tags",
        "Type": "Raw",
        "Value": [
          "SGD_CERT_VERSION",
          "SGD_CERT_SERIAL",
          "SGD_CERT_SUBJECT",
          "SGD_CERT_SIGNATURE_ALGORITHM"
        ]
      }
    ]
  }
```

C.2.2 证书解析


```
{
  "Data": [
    {
      "Name": "Certificate",
      "Type": "Raw",
      "Value": "-----BEGIN CERTIFICATE-----\r\nMIIBvDCCAWCgAwIBAgIQAKlhOmMvNjGlODv/J85DbzAMBggqgRzPVQGDdQUAMBIx\r\nEDAOBgNVBAMMB1RFU1RfQ0EwHhcNMjIwNDI1MDEyNjQ5WhcNMjMwNDI2MDEyNjQ5\r\nWjApMRowGAYDVQQDDBF0ZXN0LXhtbC1zZWN1cml0eTELMAkGA1UEBhMCQ04wWTAT\r\nBgcqhkjOPQIBBggqgRzPVQGCLQNCAATyiMNaEFQ2Ouqv8U96l2L/M3gryRlMaPKw\r\nX0car4UORTSlSwspEZD4qxztZwO6dxlhuxfOoew+fSsoBYWHe70jo38wfTAMBgNV\r\nHRMBAf8EAjAAMB0GA1UdDgQWBBTCpNmn2Q9xCFta837K55IHGmRkxDAOBgNVHQ8B\r\nAf8EBAMCBLAwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMBMB8GA1UdIwQY\r\nMBaAFP/DVbzgNvRAQHSK+Ed8Htav4mfkMAwGCCqBHM9VAYN1BQADSAAwRQIgUTDy\r\nJSaEDUpFXKir5zRvEIHPZE+5joDU6fMuUqn/MK4CIQCbi/bbAXPjY3BxD//Zgz/N\r\nYZziTdBcK4+3Jc/gKazeLA==\r\n-----END CERTIFICATE-----\r\n"
    },
    {
      "Name": "Tags",
      "Type": "Raw",
      "Value": [
        "SGD_CERT_ISSUER",
        "SGD_CERT_SUBJECT"
      ]
    }
  ]
}

{
  "Status": {
    "Code": "0",
    "Msg": "success"
  },
  "Data": [
    {
      "Name": "SGD_CERT_ISSUER",
      "Type": "Raw",
      "Value": "CN=TEST_CA"
    },
    {
      "Name": "SGD_CERT_SUBJECT",
```

```
        "Type": "Raw",
        "Value": "CN=test-xml-security,C=CN"
      }
    ]
}
```

参 考 文 献

[1] IETF RFC 7540 Hypertext Transfer Protocol Version 2 (HTTP/2)
[2] IETF RFC 8259 The JavaScript Object Notation (JSON) Data Interchange Format

中华人民共和国密码
行业标准
密码应用 HTTP 接口规范
GM/T 0136—2024
*
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)
网址 www.spc.net.cn
总编室:(010)68533533　发行中心:(010)51780238
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 2.5　字数 61 千字
2025 年 6 月第 1 版　2025 年 6 月第 1 次印刷
*
书号:155066·2-39070　定价 65.00 元

如有印装差错　由本社发行中心调换
版权专有　侵权必究
举报电话:(010)68510107

GM/T 0136-2024