ICS 35.240.99
CCS L 80

团 体 标 准

T/ZJCX 0047—2024

浙江省法人数字证书应用接口规范

Application Interface Specification for
Legal-person Entity Digital Certificate of
Zhejiang Province

2024 -12- 24 发布　　2024 -12- 25 实施

浙江省企业技术创新协会　发布

目　　次

前　言

本文件按照 GB/T1.1-2020《标准化导则 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。

本文件由浙江省技术创新服务中心提出。

本文件由浙江省企业技术创新协会归口。

本文件主要起草单位：浙江省数字安全证书管理有限公司。

本文件参与起草单位：浙江经信信息技术中心有限公司、浙江省数据安全服务有限公司、浙江华东云网络安全中心、浙江东安检测技术有限公司。

本文件主要起草人：金楠洋、徐强、李红乾、方伟明、朱世杰、郭斌超、秦盼春、林方中、贾晓燕、程动平、黄子烨、蔡金兰、夏焱、王飞、柴叶蓉、郭敏、陆琳、胡燕雄。

引　言

信息与网络技术在促进社会经济、科技、文化、教育和管理等各个方面发展的同时，也带来了信息安全风险。随着电子政务业务的快速发展和应用的日益增多，需要在电子政务网络环境中建立真实、有效的身份信任体制，确认电子政务业务参与方的有效身份，建立彼此间的信任关系及保证信息的真实性、完整性、机密性和操作的不可否认性。

本文件根据《中华人民共和国电子签名法》《电子认证服务管理办法》《电子政务电子认证服务管理办法》的要求，以国家密码管理局电子政务数字证书格式规范为基础，兼容人力资源和社会保障电子认证数字证书格式规范等行业标准的要求，根据浙江省法人数字证书的具体要求，起草本文件以保证浙江省法人数字证书可在浙江省内适用该证书的应用系统范围内统一、规范、通用。在本文件实施过程中同时遵守其他国家相关法律法规。

浙江省法人数字证书应用接口规范

1 范围

本文件规定了浙江省法人数字证书应用接口的规范性引用文件、术语和定义、缩略语、接口类型、密码设备及标识、接口功能、接口协议、数据格式、接口常量值和接口服务支持。

本文件适用于基于接口调用证书应用模式的应用系统开发与集成，包括通过签名及认证服务器或安全应用支撑平台实现数字证书身份认证、消息摘要、数字签名和验证、签名数据、数字信封等证书应用功能。

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，仅所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（包括所有的修订单）适用于本文件。

GM/T 0003.2 SM2椭圆曲线公钥密码算法 第2部分：数字签名算法

GM/T 0004 SM3密码杂凑算法

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 0015 数字证书格式

GM/T 0016 智能密码钥匙密码应用接口规范

GM/T 0017 智能密码钥匙密码应用接口数据格式规范

GM/T 0027 智能密码钥匙技术规范

GM/Z 0001 密码术语

3 术语和定义

下列术语和定义适用于本文件。

3.1

数据 DATA

任意类型的字节串。

3.2

数字证书 Digital Certificate

根据GM/Z 0001《密码术语》，数字证书也称公钥证书，由证书认证机构（CA）签名的包含公开密钥拥有者信息、公开密钥、签发者信息、有效期以及扩展信息的一种数据结构。按类别可分为个人证书、机构证书和设备证书，按用途可分为签名证书和加密证书。

3.3

签名证书 Signature Certificate

用于证明签名公钥的数字证书。

3.4

加密证书 Encipherment Certificate/Exchange Certificate

用于证明加密公钥的数字证书。

3.5

浙江省法人数字证书 Legal-person Entity Digital Certificate of Zhejiang Province

浙江省法人数字证书是浙江省法人数字证书管理机构通过政府购买服务的方式，由依法取得电子政务电子认证服务机构资质的机构为浙江省党政机关、社会团体、企事业单位法人和个体工商户等提供的数字证书。

3.6

数字签名 Digital Signature

签名者使用私钥对待签名数据的杂凑值做密码运算得到的结果，该结果只能用签名者的公钥进行验证，用于确认待签名数据的完整性、签名者身份的真实性和签名行为的抗抵赖性。

3.7

验证签名 Verify Signature

验证签名通常指的是验证数字签名的有效性，以确保信息的真实性和未被篡改。这一过程中，接收方会使用发送方的公钥来解密数字签名，并对比解密后的摘要与原始信息的摘要是否一致。

3.8

加密 Encipherment/Encryption

对数据进行密码变换以产生密文的过程。

3.9

解密 Decipherment/Decryption

加密过程对应的逆过程。

3.10

对称密码算法 Symmetric Cryptographic Algorithm

加密和解密使用相同密钥的密码算法。

3.11

非对称密码算法 Asymmetric Cryptographic Algorithm

加密和解密使用不同密钥的密码算法。其中一个密钥（公钥）可以公开，另一个密钥（私钥）必须保密，且由公钥求解私钥是计算不可行的。

3.12

消息摘要 Message Digest

消息经过密码杂凑运算得到的结果。

3.13

杂凑值 Hash Value

密码杂凑运算的结果。

3.14

密码设备 Cryptographic Device

实现密码运算、密钥管理等功能，提供密码服务的设备。

3.15

密码设备事件 Cryptographic Device Event

指在密码设备生命周期内，由于各种原因导致的设备状态变化、操作执行等与密码设备相关的事件。

3.16

智能密码钥匙 Cryptographic Smart Token

实现密码运算、密钥管理功能，提供密码服务的终端密码设备，一般使用USB接口形态，又称“USBKEY”。

3.17

密钥容器 Key Container

密码设备中用于保存密钥所划分的唯一性存储空间。

3.18

数字信封 Digital Envelope

一种数据结构，包含用对称密钥加密的密文和用公钥加密的该对称密钥。

4 缩略语

SSL 安全套接层协议（Secure Socket Layer）

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

CRL 证书吊销列表（Certificate Revocation List）

CA 证书认证机构（Certificatation Authority）

DER 可区分编码规则（Distinguished Encoding Rules）

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

OCSP 在线证书状态协议（Online Certificate Status Protocol）

OID 对象标识符（Object Identifier）

USBKEY 智能密码钥匙，具有 USB 接口的硬件密码设备（USB Key）

5 接口类型

浙江省法人数字证书应用接口驻留在浙江省内适用该证书的应用系统服务器端或客户端内部，由应用系统调用。应用接口分为以下两个大类：

1）服务器端接口

服务器端接口集成在应用系统业务服务器内，以安全方式为应用系统提供身份认证、消息摘要、数字签名和验证签名、签名数据生成和验证、数字信封封装和解开的功能。

2）客户端接口

客户端接口集成在应用系统客户端，为 B/S 结构和 C/S 结构系统的客户端提供消息摘要、数字签名和验证签名、签名数据生成和验证、数字信封封装和解开功能。

6 密码设备及标识

6.1 密码设备

密码设备中可容纳多对公私钥和多个证书，须使用密钥容器名称标识所使用的密钥和证书。

应用接口支持的密码设备类型包括：

1）USBKEY

USBKEY 在客户端使用，必须支持 GM/T 0016 《智能密码钥匙密码应用接口规范》，且采用支持国产密码算法的、获得商用密码产品认证的产品。

USBKEY 内部的设备类型编码应设置为 0x0001；

2）签名验签服务器

签名验签服务器在服务端使用，提供基于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。签名验签服务器必须采用支持国产密码算法的、获得商用密码产品认证的密码设备。

签名验签服务器内部的设备类型编码应设置为 0x0604。

3）身份认证服务器

身份认证服务器在服务端使用，提供基于数字证书身份认证功能的服务器，可以确保只有授权用户可以访问受保护的资源和服务。身份认证服务器必须采用支持国产密码算法的、获得商用密码产品认证的密码设备。

签名验签服务器内部的设备类型编码应设置为 0x0614。

6.2 密码设备标识

应用系统使用密码设备时，应设置密码设备的名称。按下述规则为每种密码设备设置名称，命名规则如下：

1）USBKEY命名应与注册的USBKEY名称一致。

2）对于签名验签服务器和身份认证服务器，命名应与接口模块名称一致。

7 接口功能

1）身份认证：使用密码技术对通信双方实行基于数字证书的身份认证。

2）消息摘要：对数据进行摘要运算。

3）数字签名与验证：实现数字签名和验证签名。

4）签名数据：实现签名数据格式、签名数据生成流程、签名数据验签流程。

5）数字信封：实现基于数字信封的数据加密和解密。

7.1 身份认证

每个用户都持有可唯一标识用户身份的私钥和由 CA 签发的公钥证书，用户私有信息保存在智能密码钥匙或其他安全载体中。在访问应用系统时，用户出示身份证明（签名和证书），实现基于证书的身份认证过程。

身份认证可采用单向或双向认证方式。

单向认证方式：应用系统中，应用服务程序直接通过应用接口调用认证服务器对用户进行身份认证，用户身份信息由应用系统自身进行维护和管理。

双向认证方式：应用系统服务器端和客户端需要相互认证，才能完成认证过程。

下面对双向认证流程进行描述：

1）客户端请求

用户插入智能密码钥匙，进行智能密码钥匙保护口令验证。

客户端生成随机数，将该随机数及用户 ID 上传至服务器，请求登录系统。

2）服务器应答

认证服务器用自己的私钥对客户端生成的随机数签名，并生成服务器随机数，将数字签名、服务器随机数和认证服务器证书返回给客户端。

3）客户端认证服务器端

客户端在收到应答后，检验认证服务器证书的有效性，再验证认证服务器的签名。若签名验证不正确，则本次认证过程中止，否则用用户私钥对服务器随机数签名，连同用户证书组成认证信息上传至认证服务器。

4）认证服务器认证客户端

认证服务器先验证用户证书的有效性，再验证用户的签名。签名正确，客户端身份认证成功，允许用户进行正常的业务操作，否则，用户请求被拒绝。

7.2 消息摘要

消息摘要（即HASH运算）对数据流（DATA）进行填充、迭代压缩和输出选裁，生成一个摘要结果，用于对数据的完整性验证。本文件使用的HASH算法为国家密码管理主管部门审核批准的密码杂凑算法，如SM3密码杂凑算法。

SM3密码杂凑算法是国密标准GM/T 0004《SM3密码杂凑算法》中的HASH算法。用DATA记需要进行HASH运算的数据，数据长度不限，用SM3(…)记SM3算法，用H记运算结果，则SM3运算记为：

H = SM3(DATA)

SM3运算结果H为32字节（256bit）长度数据。

7.3 数字签名与验证

7.3.1 签名算法

签名算法应符合国家密码主管部门对密码算法的规定，并根据国家密码主管部门批准的最新算法及时调整，如SM2密码算法。

7.3.2 数字签名生成流程

被签名的数据为字节串（DATA）。

1）使用 SM3 摘要算法对需要签名的字节串（DATA）进行 HASH 运算，使用 SM3 算法计算签名摘要的过程参见 GM/T 0009 《SM2 密码算法使用规范》：

H = HASH(DATA)

H 长度为 32 字节。

2)使用私钥(记作 Pvk)对摘要 H 进行签名运算，使用 SM2 算法计算签名的过程参见 GM/T 0003.2 《SM2 椭圆曲线公钥密码算法 第 2 部分：数字签名算法》，生成签名结果：

Sig = Pvk(H)

Sig 长度为 64 字节。

7.3.3 验证数字签名流程

签名结果记为 Sig，签名原数据记为 DATA，验证签名公钥为 Pbk，则验证签名的流程为：

1）使用 Pbk 对 Sig 做公钥解密运算：

H' = Pbk(Sig)

2）对签名原数据做 HASH 运算：

H = HASH(DATA)

3）比较H与H'，如相等，则验证正确，否则验证错误。

7.4 签名数据

7.4.1 签名数据格式

签名数据格式由版本号、算法标识、被签名内容信息、证书、CRLs 和签名者信息组成，编码格式为 DER。详见表 1。

表 1 签名数据格式表

序号	属性	建议值
1	版本号	1
2	算法标识	1.2.156.10197.1.501
3	被签名内容信息	数据类型
		被签名内容（可选）
4	证书	签名者证书和签发者证书（可选）
5	CRLs	证书撤销列表（可选）
6	签名者信息	版本号（1）
		证书序列号
		信息摘要算法标识

		签名时间（可选）
		签名算法标识
		签名值

在数字签名结果中可以包含被签名数据或不包含被签名数据；在数字签名结果中可以包含签名者证书也可以不包含签名者证书。

签名值是通过私钥对被签名内容和其他可认证属性进行签名运算的结果。

含多个签名值时，每个签名者可对上述签名格式中的证书、CRLs、被签名内容等可以从已生成的签名结果中删除或重新插入，自行决定对上述签名格式中的信息项的签名范围。

签名格式根据下面的需要进行选择：

1）需要保存在数据库中的数字签名与被签名原数据采用分离的方式进行存放，即在“被签名内容信息”内不含实际的“被签名内容”，被签名内容可存放在其他地方。

2）不包含签名者证书，可以减少签名数据大小，在验证时需要查找证书。

3）需要在网络上传输（例如与其他系统交换数据）的数字签名，在签名包中可以包含被签名数据和签名者证书。

7.4.2 签名数据生成流程

被签名的数据为字节串（DATA）。

1）使用 SM3 摘要算法对需要签名的字节串（DATA）进行 HASH 运算，使用 SM3 算法计算签名摘要的过程参见 GM/T 0009 《SM2 密码算法使用规范》：

H = HASH(DATA)

H 长度为 32 字节。

2)使用私钥(记作 Pvk)对摘要 H 进行签名运算，使用 SM2 算法计算签名的过程参见 GM/T 0003.2《SM2 椭圆曲线公钥密码算法 第 2 部分：数字签名算法》，生成签名结果：

Sig = Pvk(H)

Sig 长度为 64 字节。

3）按GM/T 0010《SM2密码算法加密签名消息语法规范》标准格式对签名进行编码，得到的结果记为signedData。

7.4.3 签名数据验签流程

设待验证的签名为 signedData，签名原数据为 DATA，签名结果记为 Sig，签名原数据记为 DATA，验证签名公钥为 Pbk，则验证签名的流程为：

1）从 signedData 中提取签名者证书序列号等信息，按证书序列号获取签名者证书并检验证书的有效性（在签名中提取证书或通过 LDAP 下载证书）；

2）使用 Pbk 对 Sig 作公钥解密运算：

H' = Pbk(Sig)

3）对签名原数据作 HASH 运算：

H = HASH(DATA)

4）比较 H 与 H'，如相等，则验证正确，否则验证错误。

7.5 数字信封

数字信封用于通信双方交换数据。发送方生成随机的报文密钥，使用报文密钥对发送内容进行对称加密，再用接收方的公钥对报文密钥加密，然后将加密的报文密钥和加密的发送内容，按照 GM/T 0010《SM2 密码算法加密签名消息语法规范》编码组成数字信封。发送方可以为发送内容附加数字签名。

7.5.1 数字信封格式

不带签名的数字信封格式见表 2，带签名的数字信封格式见表 3。

表 2 不带签名的数字信封格式表

序号	属性	建议值
1	版本号	0
2	接收者信息	版本号
		证书序列号
		对密钥的加密算法标识
		加密的报文密钥
3	加密内容信息	数据类型
		数据加密算法标识
		加密的内容

表 3 带签名的数字信封格式表

序号	属性	建议值
1	版本号	0
2	接收者信息	版本号
		证书序列号
		对密钥的加密算法标识
		加密的报文密钥
3	信息摘要算法	算法标识
4	加密内容信息	数据类型
		数据加密算法标识
		加密的内容（使用报文密钥加密）
5	证书	签名者证书和签发者证书（可选）
6	签名者信息	版本号（1）
		签名者证书序列号
		签名时间
		签名算法
		加密的签名值（使用报文密钥加密）
		其他属性

7.5.2 数字信封封装流程

1）随机生成报文密钥mk。

2）用接收方公钥对mk按GM/T 0010《SM2密码算法加密签名消息语法规范》标准进行加密。

3）使用mk对发送内容对称加密。

4）如果需要签名，则执行如下步骤：

a）对发送内容作数字签名（见数字签名过程，但不作编码）；

b）使用mk为密钥对签名进行加密。

5）按表2或表3，采用DER编码构成数字信封。

7.5.3 数字信封解开流程

1）使用接收者私钥解出 mk。

2）使用 mk 解密发送内容。

3）如果存在签名，则执行：

a）使用 mk 为密钥解出签名值；

b）验证签名。

8 接口协议

1）数字证书应采用 GM/T 0015《数字证书格式》标准。

2）数字签名、数字信封应采用 GM/T 0010《SM2 密码算法加密签名消息语法规范》标准。

3）消息摘要应采用 GM/T 0004《SM3 密码杂凑算法》标准。

4）密码设备接口应采用 GM/T 0017《智能密码钥匙密码应用接口数据格式规范》、GM/T 0027 《智能密码钥匙技术规范》。

5）数据编码格式应采用 DER 标准。

9 数据格式

应用接口使用以下几种类型数据格式（具体的类型定义与开发平台有关）：

1）4 字节整数和无符号整数

2）8 位二进制字节数组

3）描述不包含特殊格式或功能、以某种字符编码（如 UTF-8、ASCII）的普通文本字符串

4）16 进制字符串

5）Base64 编码字符串

10 接口常量值

10.1 密码设备事件标识常量值

设备事件类型和标识常量值见表 4。

表 4 设备事件类型和标识常量值表

序号	标识符	值	说明
1	EVENT_INSERTED	0x01	有新密码设备接入
2	EVENT_REMOVED	0x02	有设备移除
3	EVENT_UPDATED	0x03	整个设备列表已经更新，需要重新获取设备

10.2 非对称密码算法标识常量值

非对称算法类型和标识常量值见表 5。

表 5 非对称算法类型和标识常量值表

序号	标识符	值	说明
1	KEY_UNKNOWN	0x00000000	未知算法类型
2	KEY_SM2	0x00020100	SM2 算法

10.3 对称密码算法标识常量值

对称算法类型和标识常量值见表 6。

表 6 对称算法类型和标识常量值表

序号	标识符	值	说明
1	SYMM_SM1_ECB	0x00000101	SM1 ECB 对称加密算法
2	SYMM_SM4_ECB	0x00000401	SM4 ECB 对称加密算法
3	SYMM_SM1_CBC	0x00000102	SM1 CBC 对称加密算法
4	SYMM_SM4_CBC	0x00000402	SM4 CBC 对称加密算法

10.4 HASH 算法标识常量值

HASH 算法类型和标识常量值见表 7。

表 7HASH 算法类型和标识常量值表

序号	标识符	值	说明
1	HASH_SM3	0x00000001	SM3

10.5 密钥用途标识常量值

非对称密钥按用途分为“签名密钥”和“加密密钥”，具体定义和标识常量值见表 8。

表 8 密钥用途标识常量值表

序号	标识符	值	说明
1	KEYUSAGE_UNKNOWN	0x00000000	未知的密钥用途
2	KEYUSAGE_SIGN	0x00000001	签名密钥对
3	KEYUSAGE_ENC	0x00000002	加密密钥对

10.6 证书用途标识常量值

证书按照用途分为“签名证书”和“加密证书”，具体定义和标识常量值见表 9。

表 9 证书用途标识常量值表

序号	标识符	值	说明
1	CERT_UNKNOWN	0x00000000	未知的证书用途
2	CERT_SIGN	0x00000001	签名证书
3	CERT_ENC	0x00000002	加密证书

10.7 证书验证类型标识常量值

验证证书有效性时，可指定验证内容，具体定义和标识常量值见表 10。

表 10 证书验证类型标识常量值表

序号	标识符	值	说明
1	CERT_VERIFY_TIME	0x00000001	验证有效期
2	CERT_VERIFY_SIGNATURE	0x00000002	验证签名
3	CERT_VERIFY_CRL	0x00000004	验证 CRL
4	CERT_VERIFY_ALL	0x00000007	验证所有项

10.8 字符集编码标识常量值

对字符串进行签名/验签、加密/解密时，需要指定该字符串的编码类型。字符编码类型和标识常量值见表 11。

表 11 字符集编码标识常量值表

序号	标识符	值	说明
1	ENCODING_DEFAULT	0x00000000	使用默认设置（UTF-8 编码）
2	ENCODING_ASCII	0x00000001	ASCII 编码
3	ENCODING_UNICODE	0x00000002	UNICODE 编码
4	ENCODING_UTF8	0x00000004	UTF-8 编码

10.9 签名封装类型标识常量值

签名数据的封装类型和标识常量值见表 12。

表 12 签名封装类型标识常量值表

序号	标识符	值	说明
1	SIGN_UNKNOWN	0x00000000	未知的封装类型
2	SIGN_DATA	0x00000001	按签名数据格式封装签名
3	SIGN_ENVELOPED	0x00000002	按数字信封格式封装签名

10.10 密文封装类型标识常量值

密文数据的封装类型和标识常量值见表 13。

表 13 密文封装类型标识常量值表

序号	标识符	值	说明
1	ENVELOPED_UNKNOWN	0x00000000	未知的封装类型
2	ENVELOPED_DETACH	0x00000001	不带签名数据格式封装密文
3	ENVELOPED_ATTACH	0x00000002	带签名数据数字信封格式封装密文
4	ENVELOPED_SM2	0x00000003	仅用于 SM2 加密，输出 SM2 加密结果

10.11 签名附带属性标识

签名附带属性标识具体类型和标识常量值见表 14。

表 14 签名附带属性标识常量值表

序号	标识符	值	说明
1	SIGNATTACH_CERT	0x00000000	签名带证书
2	SIGNATTACH_MSG	0x00000001	签名带原文，对文件签名时不支持
3	SIGNATTACH_TIME	0x00000002	签名带时间戳

11 接口服务支持

11.1 证书应用服务管理

浙江省法人数字证书管理机构参与用户的应用系统安全需求分析，并指导证书应用部分的开发和实施。

应用系统明确安全需求分析后，填写《浙江省法人数字证书应用登记表》，签字后提交给应用系统主管部门进行审核，审核通过后签字盖章，正式提交给浙江省法人数字证书管理机构。

浙江省法人数字证书管理机构对用户提交的资料进行复核，审核通过后进行应用集成的指导实施。

11.2 技术支持服务

浙江省法人数字证书管理机构负责解释电子认证应用架构、接口实现、软件部署以及集成标准规范等相关问题。

附 录 A

（规范性附录）

客户端接口

A.1 接口描述

在应用系统客户端，由“法人证”证书助手客户端为系统提供安全服务。客户端与应用系统的接口是与平台无关的 ActiveX 控件（可使用 JavaScript 和 C/C++调用），适用于所有操作系统。

本接口由一组 COM 对象以及接口函数构成，接口主要实现密码设备管理、数字签名、数字信封和其他相关功能。

A.2 接口对象

客户端 ActiveX 控件定义的对象见表 A.1：

表 A.1 接口对象表

序号	定义	类名	说明
1	IZJFRZDeviceEnum	ZJFRZKeyManager.ZJFRZDeviceEnum.1	设备枚举器
2	IZJFRZDevice	ZJFRZKeyManager.ZJFRZDevice.1	设备对象
3	IZJFRZCertificate	ZJFRZKeyManager.ZJFRZCertificate.1	证书对象
4	IZJFRZSignedData	ZJFRZKeyManager.ZJFRZSignedData.1	数据签名对象
5	IZJFRZEnvelopedData	ZJFRZKeyManager.ZJFRZEnvelopedData.1	数字信封对象

A.3 接口属性

客户端 ActiveX 控件定义的属性见表 A.2：

表 A.2 接口属性表

序号	属性	说明
1	IZJFRZDeviceEnum.Count	设备个数
2	IZJFRZDevice.SN	设备序列号
3	IZJFRZDevice.Lable	设备标签
4	IZJFRZDevice.Manufacturer	设备厂商
5	IZJFRZDevice.CertificateCount	证书个数
6	IZJFRZCertificate.SN	证书序列号
7	IZJFRZCertificate.KeyType	证书算法类型
8	IZJFRZCertificate.KeyUsage	证书用途
9	IZJFRZCertificate.Issuer	颁发者信息
10	IZJFRZCertificate.Subject	主题信息
11	IZJFRZCertificate.NotBefore	有效起始时间
12	IZJFRZCertificate.NotAfter	有效截止时间
13	IZJFRZCertificate.IsRegisted	是否已注册至浏览器
14	IZJFRZSignedData.Type	数字签名类型
15	IZJFRZSignedData.Algo	数字签名算法
16	IZJFRZSignedData.LocalTime	数字签名附带的时间戳

A.4 接口方法

客户端 ActiveX 控件定义的方法见表 A.3：

表 A.3 接口方法表

序号	属性	说明

1	IZJFRZDeviceEnum.EnumDevices	枚举密码设备
2	IZJFRZDeviceEnum.get_Item	获取密码设备
3	IZJFRZDeviceEnum.AddHandler	添加事件监听函数
4	IZJFRZDeviceEnum.RemoveHandler	移除事件监听函数
5	IZJFRZDevice.VerifyPIN	校验用户PIN码
6	IZJFRZDevice.ChangePIN	修改用户PIN码
7	IZJFRZDevice.get_Certifcate	获取证书对象
8	IZJFRZDevice.Sign	签名数据
9	IZJFRZDevice.Verify	验证签名
10	IZJFRZDevice.Encrypt	加密数据
11	IZJFRZDevice.Decrypt	解密数据
12	IZJFRZCertificate.FromString	Base64编码的证书内容解析为证书对象
13	IZJFRZCertificate.ToString	证书对象解析为Base64编码输出
14	IZJFRZCertificate.IsValid	验证证书有效性
15	IZJFRZCertificate.get_IssuerNode	获取证书颁发者项中的某个字段值
16	IZJFRZCertificate.get_UserNode	获取证书主题者项中的某个字段值
17	IZJFRZCertificate.RegistToBrowser	注册到浏览器
18	IZJFRZCertificate.UnregisterFromBrowser	从浏览器反注册
19	IZJFRZSignedData.FromString	Base64编码的签名数据解析为签名对象
20	IZJFRZSignedData.ToString	签名对象解析为Base64编码输出
21	IZJFRZSignedData.get_Content	获取签名中的原文
22	IZJFRZSignedData.get_Certificate	获取签名中的证书对象
23	IZJFRZSignedData.Verify	验证签名
24	IZJFRZEnvelopedData.FromString	Base64编码的签名数据解析为密文对象
25	IZJFRZEnvelopedData.ToString	密文对象解析为Base64编码输出
26	IZJFRZEnvelopedData.Encrypt	加密数据

A.4.1 枚举密码设备

定义：EnumDevices(val type, val forceUpdate)

参数：type[数字]：忽略

forceUpdate[数字]：是否强制刷新设备列表，0－否；其他－是，默认为0

返回：无，执行失败将导致异常

说明：无

示例：

```
//枚举所有密码设备（不强制刷新）
deviceEnum.EnumDevices(0, 0);
```

A.4.2 获取密码设备

定义：get_Item(var index, var device)

参数：index[数字]：密码设备编号，从 0 开始

device[对象]：设备接口对象，用来接受设备对象实例

返回：无，执行失败将导致异常

说明：参数 device，JavaScript 需要先创建对象，才能作为该函数的参数传入

示例：

```
var selKey = new ActiveXObject("ZJCAKeyManager.ZJCADevice.1");
//获取第0 个设备对象实例
deviceEnum.get_Item(0, selKey);
```

A.4.3 添加设备事件监听函数

定义：AddHandler(var deventDisp)

参数：deventDisp[对象]：设备事件监听接口/函数

返回：无，执行失败将导致异常

说明：浏览器页面加载时创建“设备枚举器”对象，然后添加事件监听函数

示例：

```
/*
*name：发生事件的设备标签
*index：发生事件的设备序号
*type：事件类型，1：接入；2：移除
*/
function zjca_OnKeyChanged(name, index, type) {
//此处添加处理事件代码
}
// 添加事件监听函数
deviceEnum.AddHandler(zjca_OnkeyChanged);
```

A.4.4 移除设备事件监听函数

定义：RemoveHandler(var deventDisp)

参数：deventDisp[对象]：要移除的设备事件监听接口/函数

返回：无，执行失败将导致异常

说明：浏览器页面关闭前移除事件监听函数，然后释放“设备枚举器”对象

示例：

```
// 移除事件监听函数
deviceEnum.RemoveHandler(zjca_OnKeyChanged);
```

A.4.5 验证用户PIN码

定义：VerifyPIN(var type, var pin)

参数：type[数字]：PIN 码类型，目前只支持 1－用户 PIN 码

pin[字符串]：用户 PIN 码

返回：[数字]密码错误次数，执行失败将导致异常

说明：该函数在 ActiveX 组件内部弹出 PIN 输入对话框，用来输入老 PIN 码和新 PIN 码

示例：

// 校验用户PIN 码

selKey. VerifyPIN (1, "111111");

A.4.6 修改用户PIN码

定义：ChangePIN()

参数：无

返回：[数值]密码错误次数，执行失败将导致异常

说明：该函数在 ActiveX 组件内部弹出 PIN 输入对话框，用来输入老 PIN 码和新 PIN 码

示例：

// 修改用户PIN 码

selKey. ChangePIN ();

A.4.7 获取密码设备中的证书对象

定义：get_Certificate(var index, var certObj)

参数：index[数字]：要返回的证书序号，从 0 开始

certObj[对象]：证书接口对象

返回：无，执行失败将导致异常

说明：参数 certObj，JavaScript 需要先创建对象，才能作为该函数的参数传入

示例：

var cert = new ActiveXObject("ZJCAKeyManager.ZICACertificate.1");

//获取第0个证书对象实例

selkey. get_ltem(0, cert);

A.4.8 使用密码设备签名数据

定义：Sign(var alg, var data, var encode, var userID, var signType,var signFlags, var signObj)

参数：alg[数字]：签名算法类型，为 ZJCA_KEY_RSA 或 ZCA_KEY_SM2

data[字符串]：待签名的明文数据

encode[数字]：原文字符集编码

userID[字符串]：签名时使用的描述参数

signType[数字]：签名类型，具体定义见“10.9签名封装类型标识常量值”

signFlags[数字]：签名参数，具体定义见“10.11签名附带属性标识常量值”

signObj[对象]：签名数据对象，用来接受签名后的数据

返回：无，执行失败将导致异常

说明：无

示例：

// SM2 签名数据，签名为P1 格式

```
var msg = "明文数据ABCDEF";
var signObj= new ActiveXObject("ZJCAKeyManager. ZJCASignedData.1");
selKey. Sign (2, msg, 3, "1234567812345678", 1, 0, signObj);
var signature = signObj.ToString();
```

A.4.9 使用密码设备验证签名

定义：Verify(var data,var encode,var userID,var sign,var certObj)

参数：data[字符串]：待签名的明文数据

encode[数字]：原文字符集编码

sign[字符串]：签名，Base64格式编码

userID[字符串]：签名时使用的描述参数

certObj[对象]：用于验签的签名证书对象

返回：验签成功返回 true，否则返回 false，或者导致异常

说明：无

示例：

```
//验证消息的签名
var plainText = ...;
var signature = ...;
var pass = selKey. Verify(plaintext, 3, signature, "1234567812345678", signCert);
```

A.4.10 使用密码设备加密数据

定义：Encrypt(var msg,var encode,var certObj,var type,var* enveloedObj)

参数：msg[字符串]：明文数据

encode[数字]：原文字符集编码

certObj[对象]：加密证书对象

type[数字]：封装类型，具体定义见“10.10密文封装类型标识常量值”

enveloedObj：数字信封对象，用来接受密文数据

返回：无，执行失败将导致异常

说明：无

示例：

```
// 加密消息，密文为P1 格式，以字符串形式返回
var msg = "明文数据ABCDEF";
var enveloped = new ActiveXObject("ZJCAKeyManager. ZJCAEnvelopedData.1");
selKey. Encrypt (msg, 3, exchCert, 1, enveloped);
var cipherText = enveloped.ToString();
```

A.4.11 使用密码设备解密数据

定义：Decrypt(var alg, var cipher,var encode,var outStream)

参数：alg[数值]：加密算法

cipher[字符串]：密文数据

encode[数字]：解密后字符集编码

varOutStream[对象]：忽略，必须传NULL

返回：则返回密文字符串，Base64 格式

说明：无

示例：

```
// 解密 SM2消息，结果以字符串形式返回
var cipheriext= ...;
var decrypted=selKey. Decrypt (2, cipherText, 3, null);
```

A.4.12 构造证书

定义：FromString(var base64Cert)

参数：base64Cert[字符串]：证书内容，以 Base64 格式编码

返回：无，执行失败将导致异常

说明：证书内容必须为 X509 格式（文件后缀为*.cer）

示例：

```
// 由证书内容构造证书对象
var certContent = …;
var cert= new ActiveXObject("ZJFRZKeyManager. ZJFRZCertificate.1");
cert. FromString (certContent);
```

A.4.13 获取证书内容

定义：ToString()

参数：无。

返回：证书内容，以 Base64 格式编码。

说明：输出的证书内容为 X509 格式（文件后缀为*.cer）

示例：

```
// 将证书内容导出
var cerContent = cert. ToString();
```

A.4.14 验证证书有效性

定义：IsValid(var flags)

参数：flags[数字]：验证标识符，具体定义见“10.7 证书验证类型标识常量值”

返回：[数字]返回有效性验证结果，0－证书有效；否则返回一个错误代码

说明：无

示例：

```
// 验证证书的有效性
var valid = cert. IsValidType(0x07);
```

A.4.15 获取证书颁发者字段

定义：get_IssuerNode(var nodeName)

参数：nodeName[字符串]：指定要返回的字段名，比如"CN"、"C"等。如果该参数为""，则该函数返回全部字段信息，等同于属性 Issuer

返回：[字符串]返回证书使用者信息，或者是其中某个字段的值

说明：无

示例：

```
// 获取证书颁发者CN
var userCN = cert. get_IssuerNode("CN");
```

A.4.16 获取证书主题字段

定义：get_UserNode(var nodeName)

参数：nodeName[字符串]：指定要返回的字段名，比如"CN"、"C"等。如果该参数为""，则该函数返回全部字段信息，等同于属性 Subject

返回：[字符串]返回证书使用者信息，或者是其中某个字段的值

说明：无

示例：

```
//获取证书使用者CN
var userCN = cert.get UserNode("CN");
```

A.4.17 注册证书

定义：RegistToBrowser()

参数：无

返回：无，执行失败将导致异常

说明：无

示例：

```
// 将证书注册到浏览器
cert. RegistToBrowser();
```

A.4.18 注销证书

定义：UnregistToBrowser()

参数：无。

返回：无，执行失败将导致异常

说明：无。

示例：

```
// 将证书从浏览器反注册
cert. UnregistToBrowser();
```

A.4.19 构造签名对象

定义：FromString(var base64Sign)

参数：base64Sign：Base64 编码的签名数据

返回：无，执行失败将导致异常

说明：无

示例：

```
// 构造数字签名对象
var signatureText = …;
var signedData = new ActiveXObject("ZJCAKeyManager. ZJCASignedData.1");
signedData. FromString(signatureText);
```

A.4.20 获取签名结果

定义：ToString()

参数：无

返回：[字符串]Base64 编码的签名数据，执行失败将导致异常

说明：无

示例：

```
// 返回签名内容
var signatureText = signedData. ToString ();
```

A.4.21 获取签名中的原文

定义：get_Content(var encode)

参数：encode[数字]：原文字符集

返回：[字符串]签名中的原文数据，允许为空，只有 P7 格式的签名包含原文

说明：无

示例：

```
// 获取数字签名对象中的原文
var plainText = signedData.get_Content()
if (plaintext) {
//签名含原文
}
else {
```

//证书不含原文

}

A.4.22 获取签名中的证书对象

定义：get_Certificate(var certObj)

参数：certObj：证书对象

返回：无，执行失败将导致异常

说明：P7 格式的签名具有证书数据

示例：

```
//获取数字签名对象中的证书对象
var signCert = new ActiveXObject("ZJCAKeyManager. ZJCACertificate.1");
signedData. get_Certificate (signCert);
```

A.4.23 软件验证签名

定义：Verify(var data,var encode,var userID,var certObj)

参数：data[字符串]：待签名的明文数据

encode[数字]：原文字符集编码

返回：[布尔类型]验签成功返回 true，否则返回 false，或者导致异常

说明：无

示例：

```
// 验证消息的签名
var plainText = …;
var signature = …;
var pass = signedObj. Verify(plaintext, 3, "1234567812345678", signCert);
```

A.4.24 构造密文对象

定义：FromString(ver base64Cipher)

参数：base64Cipher[字符串]：Base64 编码的密文数据

返回：无，执行失败将导致异常

说明：无

示例：

```
// 构造数字信封对象
var cipherText = …;
var envelopedData = new ActiveXObject("ZJCAKeyManager. ZJCAEnvelopedData.1");
envelopedData. FromString(cipherText);
```

A.4.25 获取密文结果

定义：ToString()

参数：无

返回：[字符串]返回的 Base64 编码的密文数据，执行失败将导致异常

说明：无

示例：

```
// 返回密文字符串
var cipherText = envelopedData. ToString ();
```

A.4.26 软件加密

定义：Encrypt(var msg,var encode,var certObj,var type)

参数：msg[字符串]：明文数据

encode[数字]：原文字符集编码

certObj[对象]：加密证书对象

type[数字]：封装类型，具体定义见“10.10密文封装类型标识常量值”

返回：无，执行失败将导致异常

说明：无

示例：

```
// 软件加密消息，密文为P1  格式，以字符串形式返回
var msg = "明文数据ABCDEF";
var enveloped = new ActiveXObject("ZJCAKeyManager. ZJCAEnvelopedData.1");
enveloped . Encrypt (msg, 3, exchCert, 1);
var cipherText = enveloped.ToString();
```

附 录 B

（规范性附录）

客户端 Javascript 包接口

B.1 接口描述

客户端 JavaScript 包接口是一组*.JS 文件，提供了对客户端接口函数的封装，提供了一组简洁明了的常用功能函数，可供应用系统的 HTML 语言调用。应用系统可直接调用该 JavaScript 包接口函数，不用直接调用客户端接口，从而减轻应用系统的开发工作量，达到快速开发应用系统的目的。

该 JavaScript 接口适用于 Windows 操作系统，支持目前所有主流的浏览器，比如 IE、Edge、Chrome、Firefox 以及 360 浏览器等。

B.2 接口对象

客户端 JavaScript 包定义的对象见表 B.1：

表 B.1 接口对象表

序号	定义	对象名	说明
1	ZJFRZ_KEY	密码设备属性对象	获取密码设备的常用属性
2	ZJFRZ_Cert	证书属性对象	获取证书的常用属性
3	ZJFRZ_COM	ActiveX接口对象	对ActiveX控件接口的封装
4	ZJFRZ_Websocket	Websocket接口对象	对Websocket接口的封装

B.3 接口属性

客户端 JavaScript 包定义的属性见表 B.2：

表 B.2 接口属性表

序号	属性	说明
1	ZJFRZ_Key.getIndex	获取设备索引号
2	ZJFRZ_Key.getSN	获取设备序列号
3	ZJFRZ_Key.getLabel	获取标签
4	ZJFRZ_Key.getManufacturer	获取设备制造商
5	ZJFRZ_Cert.getIndex	获取证书索引号
6	ZJFRZ_Cert.getSN	获取证书序列号
7	ZJFRZ_Cert.getAlg	获取证书算法类型
8	ZJFRZ_Cert.getUsage	获取证书用途
9	ZJFRZ_Cert.getDN	获取主题项
10	ZJFRZ_Cert.getSubjectCN	获取主题项中的CN项
11	ZJFRZ_Cert.getIssuer	获取颁发者信息项
12	ZJFRZ_Cert.getIssuerCN	获取颁发者信息项中的CN项
13	ZJFRZ_Cert.getNotBefore	获取有效期起始时间
14	ZJFRZ_Cert.getNotAfter	获取有效截止时间
15	ZJFRZ_Cert.getKeySN	获取证书所在的密码设备序列号

B.4 接口方法

该 JavaScript 包接口定义的函数见表 B.3：

表 B.3 接口方法表

序号	函数	说明
1	init	初始化函数
2	close	结束函数
3	getKeyList	获取密码设备列表
4	getCertList	获取证书列表
5	getCertContent	获取证书内容
6	signMessage	签名消息
7	verifyMessage	验证消息签名
8	encryptMessage	加密消息
9	decryptMessage	解密消息

B.4.1 初始化函数

定义：init()

参数：无

返回：[布尔类型]：成功返回 true，否则返回 false

说明：无

B.4.2 结束函数

定义：close()

参数：无

返回：无

说明：无

B.4.3 获取密码设备列表

定义：getKeyList()

参数：无

返回：[数组]一个 zjca_Key 类型的数组

说明：无

B.4.4 获取证书列表

定义：getCertList(usage)

参数：usage[数字]：证书用途类型

返回：[数组]一个 zjca_Cert 类型的数组

说明：无

B.4.5 获取证书内容

定义：getCertContent(cert)

参数：cert[对象]：一个 zjca_Cert 类型的对象

返回：[字符串]Base64 编码的证书内容

说明：无

B.4.6 签名消息

定义：signMessage(cert, type, flags, message)

参数：cert[对象]：签名证书，一个 zjca_Cert 类型的对象

type[数字]：签名封装类型

flags[数字]：签名附带属性标识，仅当 type=2 时有效

message[字符串]：原文消息

返回：[字符串]Base64 编码的签名结果

说明：无

B.4.7 验证消息签名

定义：verifyMessage(key, message, base64Sign, base64Cert)

参数：key[对象]：一个 zjca_Key 类型对象，如果为 null 则使用软件验证

message[字符串]：原文消息

base64Sign[字符串]：Base64 编码的签名

base64Cert[字符串]：Base64 编码的签名证书

返回：[布尔类型]成功返回 true；否则返回 false

说明：无

B.4.8 加密消息

定义：encryptMessage(key, message, base64Cert, type)

参数：key[对象]：一个 zjca_Key 类型对象，如果为 null 则使用软件加密

message[字符串]：原文消息

base64Cert[字符串]：Base64 编码的加密证书

type[数字]：密文封装类型

返回：[字符串]Base64 编码的密文结果

说明：无

B.4.9 解密消息

定义：decryptMessage(cert, cipher)

参数：cert[对象]：加密证书，一个 zjca_Cert 类型对象

cipher[字符串]：Base64 编码的密文

返回：[字符串]解密后的原文

说明：无

附 录 C

（规范性附录）

服务器端接口

C.1 接口描述

在应用系统服务器端，由身份认证系统和签名验证服务器为系统提供服务。签名验证及身份认证服务与应用系统的接口是与平台无关的 JAVA 接口，适用于所有操作系统。

本接口由一组 JAVA 函数构成，接口主要实现身份认证、数字签名验证和其他相关功能。

C.2 接口属性

接口属性见表 C.1：

表 C.1 接口属性表

序号	定义	说明
1	int	返回的错误码
2	String	返回对应错误码的错误信息
3	RawVerifySignRequest	签名验证请求对象
4	VerifySignResponse	验签结果描述类，验签的结果不单单是返回被验证的签名是否有效
5	AuthClientResponse	身份认证结果描述类，身份认证结果的详细信息

C.3 接口函数

函数列表见表 C.2:

表 C.2 接口函数表

序号	定义	说明
1	verifyRawSign	验证PKCS#1格式签名
2	genAuthCode	获取用户登录挑战码
3	auth	验证客户端身份信息

C.3.1 验证PKCS#1 格式签名

```
public VerifySignResponse verifyRawSign(byte[] sign, byte[] cert,
    byte[] source, boolean isPlaintext, String charset, String algorithm)
    throws UnsupportedCommException, UnknownHostException, IOException,
    DocumentException, UnsupportedCommandException, VerifySignException;
```

功能：验证 PKCS#1 格式签名

输入：sign：签名值

cert：证书

source：原文

isPlaintext：是否是明文

charset：原文字符集

algorithm：签名算法

输出：[VerifySignResponse]签名验证结果

异常：VerifySignException：签名验证发生异常

UnsupportedCommandException：不支持的指令

DocumentException：文件格式异常

IOException：通讯链路异常

UnknownHostException：知的签名验证服务地址

UnsupportedCommException：只支持的通讯协议

C.3.2 获取用户登录挑战码

public static String genAuthCode(ServerConfig config, ClientInfo clientInfo)
throws DocumentException, UnsupportedCommandException,
BizException, ServerIsTooBusyException, Throwable;

功能：获取用户登录挑战码

输入：config：服务端配置

clientInfo：客户端配置

输出：[String]用户登录挑战码

异常：BizException：业务处理发生异常

UnsupportedCommandException：不支持的指令

DocumentException：文件格式异常

ServerIsTooBusyException：服务端繁忙

Throwable：未知的异常

C.3.3 验证客户端身份信息

public static AuthClientResponse auth(final String authCode,
final String cert, final String signature, final String signAlgo,
final String charset, ServerConfig config, ClientInfo clientInfo)
throws DocumentException, UnsupportedCommandException,
BizException, ServerIsTooBusyException, Throwable;

功能：验证客户端身份信息

输入：authCode：用户登录挑战码

cert：签名证书，Base64编码

signature：签名值

signAlgo：签名算法

charset：字符集

config：服务端配置

clientInfo：客户端配置

输出：[AuthClientResponse]用户身份认证结果

异常：BizException：业务处理发生异常

UnsupportedCommandException：不支持的指令

DocumentException：文件格式异常

ServerIsTooBusyException：服务端繁忙

Throwable：未知的异常

附　录　D

（资料性附录）

浙江省法人数字证书应用登记表

请选择业务类型，在对应的栏目中打“√”
□应用接入　　□应用变更　　□应用停用
应用信息
应用名称：_____　　主管部门：_____ 所在省份/城市：_____ 通信地址：_____　　邮政编码：□□□□□□ 统一社会信用代码：□□□□□□□□□□□□□□□□□□
联系人信息
姓　　名：_____ 职　　务：_____ 电子邮件：_____ 联系电话：_____
以下由审核单位填写
应用主管部门（盖章） 审核人（签字）：　　日期：_____年____月____日
浙江省法人数字证书管理机构（盖章） 审核人（签字）：　　日期：_____年____月____日