ICS 35.040
L 80

中华人民共和国国家标准

GB/T 25056—2010

信息安全技术　证书认证系统密码及其相关安全技术规范

Information security techniques—Specifications of cryptograph and related security technology for certificate authentication system

2010-09-02 发布　　2011-02-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

本标准附录A、附录B、附录C、附录D均为资料性附录。

本标准由国家密码管理局提出。

本标准由全国信息安全标准化技术委员会(SAC/TC 260)归口。

本标准主要起草单位：长春吉大正元信息技术股份有限公司、国家密码管理局商用密码研究中心、国家信息安全工程技术研究中心。

本标准相关参与起草单位：无锡江南信息安全工程技术中心、上海格尔软件股份有限公司、北京信安世纪科技有限公司、济南得安计算机技术有限公司、北京创原天地科技有限公司、卫士通信息产业股份有限公司、天津市国瑞数码安全系统有限公司、兴唐通信科技股份有限公司、中国科学院数据与通信保护研究教育中心、北京格方网络技术有限公司、北京天融信科技有限公司、维豪信息技术有限公司等。

本标准主要起草人：邱泽军、王永传、何立波、谢永泉、姜玉琳、刘海龙、邓开勇、罗鹏、田景成、赵丹、张文建、李大为。

袁文恭、刘平、何良生、邱钢、陈连俊等专家指导了本标准的起草。

信息安全技术　证书认证系统密码及其相关安全技术规范

1　范围

本标准规定了为公众服务的数字证书认证系统的设计、建设、检测、运行及管理规范。本标准为实现数字证书认证系统的互连互通和交叉认证提供统一的依据，指导第三方证书认证机构的数字证书认证系统的建设和检测评估，规范数字证书认证系统中密码及相关安全技术的应用。

本标准适用于第三方证书认证机构的数字证书认证系统的设计、建设、检测、运行及管理。非第三方证书认证机构的数字证书认证系统的建设、运行及管理，可参照本标准。

2　规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

GB/T 2887—2000　电子计算机场地通用规范

GB/T 9361—1988　计算机场地安全要求

GB/T 20518—2006　信息安全技术　公钥基础设施　数字证书格式

GB 50174—2008　电子信息系统机房设计规范

SJ/T 10796—1996　计算机机房用活动地板技术条件

3　术语和定义

下列术语和定义适用于本标准。

3.1

认证机构证书　authority certificate

签发给证书认证机构的证书。

3.2

CA 证书　CA certificate

由一个 CA 给另一个 CA 签发的证书，一个 CA 也可以为自己签发证书，这是一种自签名的证书。

3.3

证书认证系统　certificate authentication system

对生命周期内的数字证书进行全过程管理的安全系统。

3.4

证书策略　certificate policy

是一个指定的规则集合，它指出证书对于具有普通安全需求的一个特定团体和(或)具体应用类的适用性。例如，一个特定的证书策略可以指出一个类型的证书对在一定的价格幅度下商品交易的电子数据处理的认证的适用性。

3.5

证书撤销列表　certificate revocation list；CRL

标记一系列不再被证书发布者所信任的证书的签名列表。

3.6

证书验证 certificate validation

确定证书在指定的时间内是否有效的过程。证书验证包括有效期验证、签名验证以及证书状态的检验。

3.7

证书认证机构 certificate authority;CA

又称为认证中心或CA,它是被用户所信任的签发公钥证书及证书撤销列表的管理机构。

3.8

CA注销列表 certificate authority revocation list;ARL

标记已经被注销的CA的公钥证书的列表,表示这些证书已经无效。

3.9

证书认证路径 certification path

在目录信息树中对象证书的一个有序的序列。路径的初始节点是最初待验证对象的公钥,可以通过路径获得最终的顶点的公钥。

3.10

证书撤销列表分布点 certificate revocation list distribution point

一个目录条目或其他的证书撤销列表分布源,一个通过证书撤销列表分布点发布的证书撤销列表,可以包括由一个CA发布的所有证书中的一个证书子集的注销条目,也可以包括全部证书的注销条目。

3.11

增量证书撤销列表 delta-CRL;dCRL

是一个部分的证书撤销列表,它只包括那些在基础证书撤销列表确认后注销状态改变的证书的条目。

3.12

证书序列号 certificate serial number

在一个证书认证机构所签发的证书中用于唯一标识数字证书的一个整数。

3.13

数字证书 digital certificate

由证书认证机构签名的包含公开密钥拥有者信息、公开密钥、签发者信息、有效期以及一些扩展信息的数字文件。

3.14

完全的证书撤销列表 full CRL

在给定的范围内,包含所有已经被注销的证书的证书撤销列表。

3.15

私钥 private key

在公钥密码系统中,用户的密钥对中只有用户本身才能持有的密钥。

3.16

公钥 public key

在公钥密码系统中,用户的密钥对中可以被其他用户所持有的密钥。

3.17

证书注册机构 registration authority;RA

证书认证体系中的一个组成部分,它是接收用户证书及证书撤销列表申请信息、审核用户真实身份、为用户颁发证书的管理机构。

3.18

依赖方 relying party

使用证书中的数据进行决策的用户或代理。

3.19

安全策略 security policy

由证书认证机构发布的用于约束安全服务以及设施的使用和提供方式的规则集合。

3.20

信任 trust

通常说一个实体信任另一个实体表示后一个实体将完全按照第一个实体的规定进行相关的活动。在本标准中，信任用来描述一个认证实体与证书认证机构之间的关系。

4 缩略语

ARL	Certificate Authority Revocation List CA 注销列表
CA	Certificate Authority 证书认证机构
CRL	Certificate Revocation List 证书撤销列表
dCRL	Delta-CRL 增量证书撤销列表
DIT	Derectory Information Tree 目录信息树系统
HTTP	Hypertext Transfer Protocol 超文本传输协议
HTTPS	Secure Hypertext Transfer Protocol 安全超文本传输协议
KMC	Key Management Centre 密钥管理中心
LDAP	Lightweight Directory Access Protocol 轻量目录访问协议
OCSP	Online Certificate Status Protocol 在线证书状态查询协议
OID	Object ID 对象标识符
RA	Registration Authority 证书注册机构
SOCSP	Simple Online Certificate Status Protocol 简明在线证书状态查询协议

5 证书认证系统

5.1 概述

证书认证系统是对生命周期内的数字证书进行全过程管理的安全系统。证书认证系统必须采用双证书(用于数字签名的证书和用于数字加密的证书)机制，并建设双中心(证书认证中心和密钥管理中心)。证书认证系统在逻辑上可分为核心层、管理层和服务层，其中，核心层由密钥管理中心、证书/证书撤销列表签发系统、证书/证书撤销列表存储发布管理系统构成；管理层由证书管理系统和安全管理系统构成；服务层由证书注册管理系统(包括远程用户注册管理系统)和证书查询系统构成。建议的证书认证系统的逻辑结构如图 1 所示。

5.2 功能描述

5.2.1 概述

证书认证系统提供了对生命周期内的数字证书进行全过程管理的功能，包括用户注册管理、证书/证书撤销列表的生成与签发、证书/证书撤销列表的存储与发布、证书状态的查询和密钥的生成与管理以及安全管理等。

5.2.2 用户注册管理系统

用户注册管理系统负责用户的证书申请、身份审核和证书下载，可分为本地注册管理系统和远程注册管理系统。

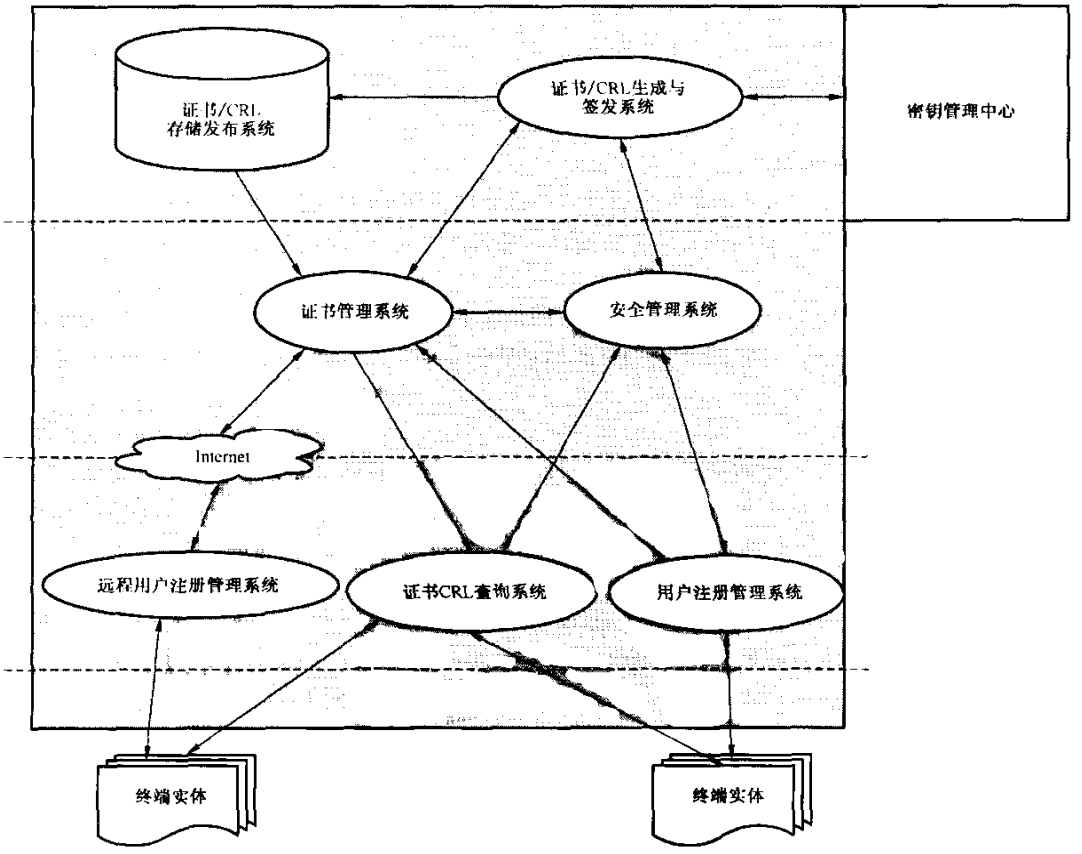

图 1　证书认证系统逻辑结构

5.2.2.1　**证书申请**

证书申请可采用在线或离线两种方式：

——在线方式：用户通过互联网等登录到用户注册管理系统申请证书；

——离线方式：用户到指定的注册机构申请证书。

5.2.2.2　**身份审核**

审核人员通过用户注册管理系统，对证书申请者进行身份审核。

5.2.2.3　**证书下载**

证书下载可采用在线或离线两种方式：

—— 在线方式：用户通过互联网等登录到用户注册管理系统下载证书；

—— 离线方式：用户到指定的注册机构下载证书。

5.2.3　**证书/证书撤销列表生成与签发系统**

5.2.3.1　**功能**

证书/证书撤销列表生成与签发系统负责生成、签发数字证书和证书撤销列表。

5.2.3.2　**证书类型**

按主体对象，证书分为人员证书、设备证书和机构证书三种类型。

按功能，证书分为加密证书和签名证书两种类型。

5.2.3.3　**证书机制**

证书认证系统采用双证书机制。每个用户拥有两张数字证书，一张用于数字签名，另一张用于数据

加密。用于数字签名的密钥对可以由用户利用具有密码运算功能的证书载体产生；用于数据加密的密钥对由密钥管理中心产生并负责安全管理。签名证书和加密证书一起保存在用户的证书载体中。

5.2.3.4 证书生成/签发

用户的数字证书由该系统的CA签发，根CA的数字证书由根CA自己签发，下级CA的数字证书由上级CA签发。

5.2.3.5 证书撤销列表

证书撤销列表是在证书有效期之内，CA签发的终止使用证书的信息，分为用户证书撤销列表(CRL)和CA证书撤销列表(ARL)两类。在证书的使用过程中，应用系统通过检查CRL/ARL，获取有关证书的状态。

5.2.4 证书/证书撤销列表存储与发布系统

证书/证书撤销列表存储与发布系统负责数字证书、证书撤销列表的存储和发布。

根据应用环境的不同，证书/证书撤销列表存储与发布系统应采用数据库或目录服务方式，实现数字证书/证书撤销列表的存储、备份和恢复等功能，并提供查询服务。

使用目录服务方式，应采用主、从目录结构以保证主目录服务器的安全，同时从目录服务器可以采用分布式的方式进行设置，以提高系统的效率。用户只能访问从目录服务器。

5.2.5 证书状态查询系统

证书状态查询系统应为用户和应用系统提供证书状态查询服务，包括：

——CRL查询：用户或应用系统利用数字证书中标识的CRL地址，下载CRL，并检验证书有效性。

——在线证书状态查询：用户或应用系统按照OCSP协议，实时在线查询证书的状态。

在实际应用中，可以根据具体情况采用上述两种查询方式之一或全部。

5.2.6 证书管理系统

证书管理系统是证书认证系统中实现对证书/证书撤销列表的申请、审核、生成、签发、存储、发布、注销、归档等功能的管理控制系统。

5.2.7 安全管理系统

安全管理系统主要包括安全审计系统和安全防护系统。

安全审计系统提供事件级审计功能，对涉及系统安全的行为、人员、时间等记录进行跟踪、统计和分析。

安全防护系统提供访问控制、入侵检测、漏洞扫描、病毒防治等网络安全功能。

5.3 系统设计

5.3.1 概述

证书认证系统的设计包括系统的总体设计和各子系统设计，本标准提供证书认证系统的设计原则以及各个子系统的实现方式，在具体实现过程中，应根据所选择的开发平台和开发环境进行详细设计。

5.3.2 总体设计原则

证书认证系统的总体设计原则如下：

a) 证书认证系统遵循标准化、模块化设计原则；

b) 证书认证系统设置相对独立的功能模块，通过各模块之间的安全连接，实现各项功能；

c) 各模块之间的通信采用基于身份鉴别机制的安全通信协议；

d) 各模块使用的密码运算都必须在密码设备中完成；

e) 各模块产生的审计日志文件采用统一的格式传递和存储；

f) 用户注册管理系统、证书/证书撤销列表生成与签发系统和密钥管理中心可以设置独立的数据库；

g) 证书认证系统的各模块应设置有效的系统管理功能；

h) 系统必须具备访问控制功能；

i) 系统在实现证书管理功能的同时，必须充分考虑系统本身的安全性。

5.3.3 用户注册管理系统设计

5.3.3.1 用户注册管理系统功能

用户注册管理系统负责用户证书/证书撤销列表的申请、审核以及证书的制作，其主要功能如下：

——用户信息的录入：录入用户的申请信息，用户申请信息包括签发证书所需要的信息，还包括用于验证用户身份的信息，这些信息存放在用户注册管理系统的数据库中。用户注册管理系统应能够批量接受从外部系统生成的、以电子文档方式存储的用户信息。

——用户信息的审核：提取用户的申请信息，审核用户的真实身份，当审核通过后，将证书签发所需要的信息提交给签发系统。

——用户证书下载：用户注册管理系统提供证书下载功能，当签发系统为用户签发证书后，用户注册管理系统能够下载用户证书，并将用户证书写入指定的用户证书载体中，然后分发给用户。

——安全审计：负责对用户注册管理系统的管理人员、操作人员的操作日志进行查询、统计以及报表打印等。

——安全管理：对用户注册管理系统的登陆进行安全访问控制，并对用户信息数据库进行管理和备份。

——多级审核：用户注册管理系统可根据需要采用分级部署的模式，对不同种类和等级的证书，可由不同级别的用户注册管理系统进行审核。用户注册管理系统应能够根据需求支持多级注册管理系统的建立和多级审核模式。

用户注册管理系统应具有并行处理的能力。

5.3.3.2 用户注册管理系统结构

用户注册管理系统有本地注册管理和远程注册管理两种方式，分别由注册管理、数据库、信息录入、身份审核、证书制作、安全管理以及安全审计几部分构成。其结构如图 2 所示。

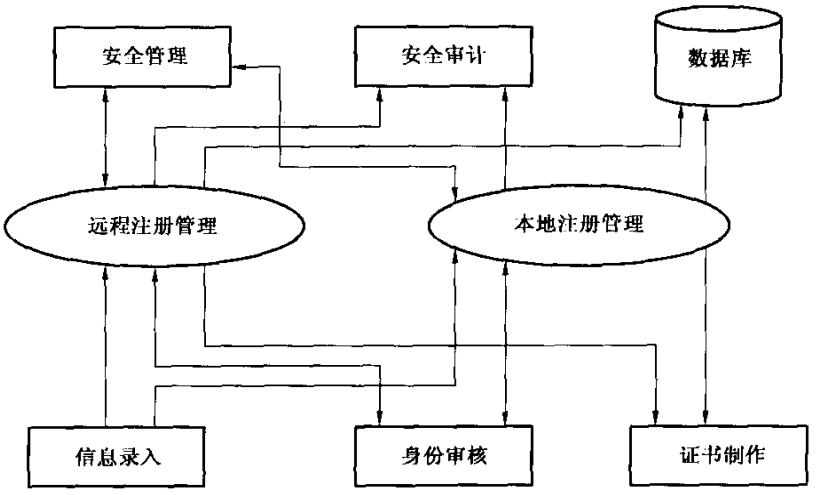

图 2 用户注册管理系统逻辑结构

5.3.4 证书/证书撤销列表生成与签发系统设计

5.3.4.1 证书/证书撤销列表生成与签发系统功能

证书/证书撤销列表生成与签发系统是证书认证系统的核心，不仅为整个证书认证系统提供签发证书/证书撤销列表的服务，还承担整个证书认证系统中主要的安全管理工作。

其主要功能如下：

—— 证书生成与签发：从数据库中读取与核对用户信息，根据拟签发的证书类型向密钥管理中心申

请加密密钥对，生成用户的签名证书和加密证书，将签发完成的证书发布到目录服务器和数据库中。根据系统的配置和管理策略，不同种类或用途的证书可以采用不同的签名密钥。

——证书更新：系统应提供CA证书及用户证书的更新功能。

——证书撤销列表生成与签发：接收注销信息，验证注销信息中的签名，然后签发证书撤销列表，将签发后的注销列表发布到数据库或目录服务器中。签发证书撤销列表的签名密钥可以与签发证书的签名密钥相同或不同。

——安全审计：负责对证书/证书撤销列表生成与签发系统的管理人员、操作人员的操作日志进行查询、统计以及报表打印等。

——安全管理：对证书/证书撤销列表生成与签发系统的登录进行安全访问控制，并对证书/证书撤销列表数据库进行管理和备份；设置管理员、操作员，并为这些人员申请和下载数字证书；配置不同的密码设备；配置不同的证书模板。

证书/证书撤销列表生成与签发系统应具有并行处理的能力。

5.3.4.2 证书/证书撤销列表生成与签发系统结构

证书/证书撤销列表生成与签发系统由证书/证书撤销列表生成与签发、安全管理、安全审计、数据库、目录服务器以及密码设备等组成。

证书/证书撤销列表生成与签发

主要功能包括证书的生成与签发和CRL/ARL的生成与签发。

a) 证书的生成/签发：根据接收的请求信息，从数据库中提取用户的信息，向密钥管理中心申请加密密钥对，然后生成并签发签名证书和加密证书，签发的证书和加密证书的私钥通过证书管理系统下传给申请者，同时将证书发布到数据库和目录服务器中。在此过程中，必须保证私钥传递的安全。

b) 证书撤销列表的生成/签发：首先验证申请信息中的数字签名和相关数据，然后签发证书撤销列表，并将注销列表发布到数据库或目录服务器指定的位置。

密码设备

密码设备完成签名以及验证工作，并负责与其他系统通信过程中的密码运算，CA的签名密钥保存在密码设备中。在进行上述工作中，必须保证所使用的密钥不能以明文形式被读出密码设备。

安全管理

主要包括：

a) 证书模板配置：不同的证书种类由不同的证书模板确定，证书模板包括相应种类证书的基本项和证书的扩展项；

b) CRL发布策略配置：配置CRL的发布策略，包括自动/人工发布模式选择、发布时间间隔；

c) 进行CA密钥的更新；

d) 进行证书的备份和归档；

e) 进行服务器安全配置，包括服务器可接受的主机访问列表；

f) 为其他子系统定义管理员以及为这些管理员签发数字证书；

g) 数据库系统的配置：数据源的选择，数据库连接的用户名和口令设置。

安全审计

查询证书/证书撤销列表生成与签发系统中的安全审计日志，并进行统计与打印。

5.3.5 证书/证书撤销列表存储发布系统设计

5.3.5.1 证书/证书撤销列表存储发布系统功能

证书/证书撤销列表存储发布系统负责证书和证书撤销列表的存储与发布，是证书认证系统的基础组成部分。证书的存储和发布必须采用数据库、目录服务器或其中之一。

该系统主要功能如下：

——证书存储；

——证书撤销列表存储；

证书和 CRL 发布；

——安全审计：负责对证书/证书撤销列表存储发布系统的管理人员、操作人员的操作日志进行查询、统计以及报表打印等；

——安全管理：对证书/证书撤销列表存储发布系统的登陆进行访问控制，并定期对数据库和目录服务器进行管理和备份；

——数据一致性检验：对数据库和目录服务器中的数据进行一致性检验。

5.3.5.2 证书/证书撤销列表存储发布系统结构

证书/证书撤销列表存储与发布系统由数据库或主/从目录服务器、安全管理模块、安全审计模块组成。

数据库

存放证书和证书撤销列表以及用户的其他信息。

目录服务器

证书/证书撤销列表存储发布系统采用主从结构的目录服务器，签发完成的数据直接写入主目录服务器中，然后由目录服务器的主从映射功能自动映射到从目录服务器中。主、从目录服务器通常配置在不同等级的安全区域。用户只能访问从目录服务器。

安全管理

主要包括：

a) 定期对数据库和目录服务器的内容进行数据的备份和归档。

b) 对数据库和目录服务器中的数据进行一致性检查，发现不一致时，应进行数据恢复。

安全审计

查询证书/证书撤销列表存储与发布系统中的安全审计日志，并进行统计与打印等。

5.3.6 证书状态查询系统设计

5.3.6.1 证书状态查询系统功能

证书状态查询系统为用户及应用系统提供证书状态查询服务。

证书状态查询系统所提供的服务可以采用以下两种方式：

——CRL 查询：用户或应用系统利用证书中标识的 CRL 地址，查询并下载 CRL 到本地，进行证书状态的检验。

——在线证书状态查询：用户或应用系统利用 OCSP 协议，在线实时查询证书的状态，查询结果经过签名后返回给请求者，进行证书状态的检验。

5.3.6.2 证书状态查询系统结构

证书状态查询系统由证书状态数据库/OCSP 服务器、安全管理模块、安全审计模块以及密码设备组成。

证书状态数据库/OCSP 服务器

接受用户及应用系统的证书状态查询请求，根据请求信息中的证书序列号，从证书状态数据库中查询证书的状态，查询结果返回给请求者。

密码设备

验证请求信息中的签名，并对查询结果进行签名。

安全管理

主要包括：

a) OCSP 服务器的配置，定义可接受的访问控制信息以及查询的证书状态数据库的地址。

b) 启动/停止查询服务，配置可接受的用户请求数量等。

安全审计

查询证书状态查询系统中的安全审计日志，并进行统计与打印等。

5.3.7 证书管理系统设计

证书管理系统是证书认证系统的综合信息控制和调度服务系统，它接收用户的各种请求信息，并将请求信息提交给相应的子系统。证书管理系统是一个逻辑上独立的系统，在进行系统设计过程中，可根据证书认证系统提供的服务，由不同的处理模块组成，这些模块可以采用分布式的结构，以增强系统的处理能力，提高系统的效率。

5.3.8 安全管理系统设计

安全管理系统主要包括安全审计系统和安全防护系统。

安全审计系统

提供事件级审计功能，对涉及系统安全的行为、人员、时间的记录进行跟踪、统计和分析。安全审计系统可以分别查询各子系统中的日志记录，也可以通过查询证书/证书撤销列表存储与发布系统中的数据库，进行集中审计。

日志记录的主要内容包括：

a) 操作员姓名；

b) 操作项目；

c) 操作起始时间；

d) 操作终止时间；

e) 证书序列号；

f) 操作结果。

日志管理的主要内容包括：

a) 日志参数设置，设置日志保存的最大规模和日志备份的目录；

b) 日志查询，查询操作员、操作事件信息；

c) 日志备份，当日志保存到日志参数设置的最大规模时，将保存的日志备份；

d) 日志处理，对日志记录的正常业务流量和各类事件进行分类整理；

e) 证据管理，对证据数据进行审计、统计和记录。

安全防护系统

提供访问控制、入侵检测、漏洞扫描、病毒防治等网络安全功能。

5.4 数字证书

关于数字证书结构和格式见 GB/T20518—2006。

其中，证书结构中的颁发者名称和主体名称的 DN 顺序应符合下列规则：

a) 如果有 C 项，则放在最后，且 C=CN；

b) 如果有 CN 项，则放在 DN 的最前面；

c) 如果同时存在 OU 和 O 项，则 OU 在 O 前面；如果同时存在 S 和 L 项，则 L 在 S 前面。

证书结构中的签名算法域中标识的密码算法必须是国家密码主管部门批准的算法。

5.5 证书撤销列表

本标准的证书撤销列表结构和格式遵照国家相关标准。

其中，证书撤销列表结构中的签名算法域中标识的密码算法必须是国家密码主管部门批准的算法。

6 密钥管理中心

6.1 结构描述

密钥管理中心由密钥生成、密钥管理、密钥库管理、认证管理、安全审计、密钥恢复和密码服务等模块组成，建议的密钥管理中心逻辑结构如图 3 所示。

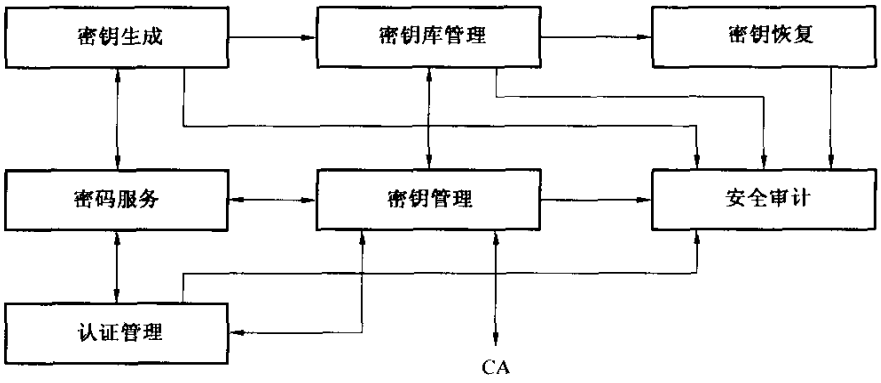

图 3 密钥管理中心逻辑结构

6.2 功能描述

6.2.1 概述

密钥管理中心提供了对生命周期内的加密证书密钥对进行全过程管理的功能，包括密钥生成、密钥存储、密钥分发、密钥备份、密钥更新、密钥撤消、密钥归档、密钥恢复以及安全管理等。

6.2.2 密钥生成

根据 CA 的请求为用户生成非对称密钥对，该密钥对由密钥管理中心的硬件密码设备生成。

6.2.3 密钥存储

密钥管理中心生成的非对称密钥对，经硬件密码设备加密后存储在数据库中。

6.2.4 密钥分发

密钥管理中心生成的非对称密钥对通过证书认证系统分发到用户证书载体中。

6.2.5 密钥备份

密钥管理中心采用热备份、冷备份和异地备份等措施实现密钥备份。

6.2.6 密钥更新

当证书到期或用户需要时，密钥管理中心根据 CA 请求为用户生成新的非对称密钥对。

6.2.7 密钥撤消

当证书到期、用户需要或管理机构依据合同规定认为必要时，密钥管理中心根据 CA 请求撤消用户当前使用的密钥。

6.2.8 密钥归档

密钥管理中心为到期或撤消的密钥提供安全长期的存储。

6.2.9 密钥恢复

密钥管理中心可为用户提供密钥恢复服务和为司法取证提供特定密钥恢复。密钥恢复需依据相关法规并按管理策略进行审批，一般用户只限于恢复自身密钥。

6.3 系统设计

6.3.1 概述

密钥管理中心的设计包括系统的整体设计和各子系统设计。本标准提供密钥管理中心的设计原则以及各个子系统的实现方式，在具体实现过程中，应根据所选择的开发平台和开发环境进行详细设计。

6.3.2 总体设计原则

a) 密钥管理中心遵循标准化、模块化设计原则；

b) 密钥管理中心设置相对独立的功能模块，通过各模块之间的安全连接，实现各项功能；

c) 各模块之间的通信采用基于身份验证机制的安全通信协议；

d) 各模块使用的密码运算都必须在密码设备中完成；

e) 各模块产生的审计日志文件采用统一的格式传递和存储；

f) 系统必须具备访问控制功能；

g) 系统应设置有效的系统管理功能；

h) 系统在实现密钥管理功能的同时，必须充分考虑系统本身的安全性；

i) 系统可为多个 CA 提供密钥服务，当为多个 CA 提供密钥服务时，由上级 CA 为密钥管理中心签发证书。

6.3.3 密钥生成模块

密钥生成模块应提供以下主要功能：

a) 非对称密钥对的生成，并将其保存在备用库中；当备用库中密钥数量不足时，自动进行补充。

b) 对称密钥的生成。

c) 随机数的生成。

6.3.4 密钥管理模块

密钥管理模块应提供以下主要功能：

a) 接收、审核 CA 的密钥申请；

b) 调用备用密钥库中的密钥对；

c) 向 CA 发送密钥对；

d) 对调用的备用密钥库中的密钥对进行处理，并将其转移到在用密钥库；

e) 对在用密钥库中的密钥进行定期检查，将超过有效期的或被撤销的密钥转移到历史密钥库；

f) 对历史密钥库中的密钥进行处理，将超过规定保留期的密钥转移到规定载体；

g) 接收与审查关于恢复密钥的申请，依据安全策略进行处理；

h) 对进入本系统的有关操作及操作人员进行身份与权限的认证。

6.3.5 密钥库管理模块

6.3.5.1 概述

密钥库管理模块负责密钥的存储管理，按照其存储的密钥的状态，密钥库分为备用库、在用库和历史库等三种类型，密钥库中的密钥数据必须加密存放。

6.3.5.2 备用库

备用库存放待使用的密钥对。密钥生成模块预生成一批密钥对，存放于备用库中；CA 需要时，可及时调出，将其提供给 CA 后转入在用库。

备用密钥库应保持一定数量的待用密钥对，存放的密钥数量依系统的用户数量而定，若少于设定的最低数量时应自动补足到规定数量。

6.3.5.3 在用库

在用库存放当前使用的密钥对。在用库中的密钥记录包含用户证书的序列号、ID 号和有效时间等标志。

6.3.5.4 历史库

历史库存放过期或已被注销的密钥对。历史库中的密钥记录包含用户证书的序列号、ID 号有效时间和作废时间等标志。

6.3.6 认证管理模块

认证管理模块负责对进入本系统的有关操作及操作人员进行身份与权限的认证。

6.3.7 安全审计模块

安全审计模块负责各个功能模块的运行事件检查、有关资料分析和密钥申请统计等服务。审计项目主要包括：

a) 运行事件记录；

b) 服务器状态记录；

c) 系统重要策略设置。

审计记录不能进行修改。

6.3.8 密钥恢复模块

密钥恢复模块负责为用户和司法取证恢复用户的加密私钥，被恢复的私钥必须安全地下载到载体。

a) 用户密钥恢复：用户通过 RA 申请，经审核后，由 CA 向密钥管理中心提出密钥恢复请求，密钥恢复模块恢复用户的密钥并通过 CA 返回 RA，下载于用户证书载体中；

b) 司法取证密钥恢复：司法取证人员必须到 KMC 进行司法取证密钥恢复，KMC 对司法取证人员的身份进行认证，认证通过后，由密钥恢复模块恢复所需的密钥并下载于特定载体中。

6.3.9 密码服务模块

密码服务模块负责为密钥管理中心的各项业务提供密码支持。

密码服务模块配置经国家密码主管部门审批的非对称密钥密码算法、对称密钥密码算法和数据摘要算法等。

密码算法必须在硬件密码设备中运行，有关密码算法、密码设备和密码接口的要求在本标准第 6 章中规定。

6.3.10 审计模块

密钥管理中心设置日志审计模块，包括全程审计和事件审计。审计员定时调出审计记录，制作统计分析表。审计员可以处理但不能修改日志审计数据。

日志记录的主要内容包括：

a) 操作员姓名；

b) 操作项目；

c) 操作起始时间；

d) 操作终止时间；

e) 证书序列号；

f) 操作结果。

日志管理的主要内容包括：

a) 日志参数设置，设置日志保存的最大规模和日志备份的目录；

b) 日志查询，日志查询主要是查询操作员、认证机构操作事件信息；

c) 日志备份，当日志保存到日志参数设置的最大规模时，将保存的日志备份；

d) 日志处理，对日志记录的正常业务流量和各类事件进行分类整理；

e) 证据管理，对证据数据进行审计、统计和记录。

6.4 KMC 与 CA 的安全通信协议

KMC 与 CA 之间采用基于身份鉴别机制的安全通信协议，并进行双向身份鉴别。

有关安全通信协议的详细内容可参见本标准 8.3“安全通信协议”。

KMC 接收来自 CA 的请求，检查确定请求合法后，为 CA 提供相应的服务，并将结果返回给 CA。KMC 与 CA 之间的消息格式参见附录 A。

7 密码算法、密码设备及接口

7.1 密码算法

证书认证系统使用对称密码算法、非对称密码算法和数据摘要算法等三类算法实现有关密码服务各项功能，其中，对称密钥密码算法实现数据加/解密以及消息认证；非对称密钥密码算法实现签名/验证以及密钥交换；数据摘要算法实现待签名消息的摘要运算。

证书认证系统使用的密码算法要求如下：

—— 对称密钥密码算法：采用国家密码主管部门批准使用的对称密码算法。

——非对称密钥密码算法：采用国家密码主管部门批准使用的非对称密钥密码算法。

——数据摘要算法：采用国家密码主管部门批准使用的数据摘要算法。数据摘要算法在实现待签名消息的摘要运算过程中，至少对部分数据要采取密码保护。

7.2 密码设备

7.2.1 概述

应采用国家密码主管部门批准使用的密码设备，包括：

——应用类密码设备：在证书认证系统中提供签名/验证、数据加密/解密、数据摘要、数字信封、密钥生成和管理等密码作业服务。

——通信类密码设备：用于KMC与CA之间、CA与RA间的传输加密。

——证书载体：具有数字签名/验证、数据加/解密等功能的智能IC卡或智能密码钥匙等载体，用于用户的证书存储及相关的密码作业。

7.2.2 密码设备的功能

密码设备必须具备如下基本功能：

a) 随机数生成；

b) 非对称密钥的产生；

c) 对称密钥的产生；

d) 非对称密钥密码算法的加解密运算；

e) 对称密钥密码算法的加解密运算；

f) 数据摘要运算；

g) 密钥的存储；

h) 密钥的安全备份和安全导入导出；

i) 多密码设备并行工作时，密钥的安全同步。

7.2.3 密码设备的安全要求

密码设备应满足下列要求：

a) 接口安全，不执行规定命令以外的任何命令和操作；

b) 协议安全，所有命令的任意组合，不能得到密钥的明文；

c) 密钥安全，密钥不以明的形式出现在密码设备之外；

d) 物理安全，密码设备应具有物理防护措施，任何情况下的拆卸均应立即销毁设备内保存的密钥。

7.3 密码服务接口

密码服务接口为调用密码服务提供统一的基本接口函数，密码设备的其他管理函数可自行定义。

本标准仅定义基本接口函数，包括：密钥对生成、非对称加解密函数、对称加解密函数、数据摘要函数等，有关函数定义以及功能说明参见附录C。

8 协议

8.1 证书管理协议

8.1.1 证书的注册申请

用户要获得证书首先必须向RA提交申请，可以采用两种申请模式：

a) 用户将自己的身份信息提交给RA，在这个过程中用户不提交自己的签名公钥；

b) 用户将自己的身份信息、签名公钥、随机选取的一段信息及签名提交给RA。

8.1.2 证书申请的审核

为用户签发证书之前，必须对用户的真实身份进行确认，要求用户提交的注册申请信息与其真实身份信息相符，同时还要验证用户拥有与签名公钥对应的签名私钥。

a) 身份确认：身份确认可以采用面对面的方式，即要求用户或其代理者携带证明资料到 RA 进行验证；也可以通过查询其他的安全应用系统的用户资料，进行自动验证。身份确认的方式需要在发布认证策略时发布。

b) 拥有签名私钥的验证：对申请信息进行数据摘要运算，然后用申请信息中的公钥，对申请信息中的签名进行解密，得到申请者计算的数据摘要，然后进行比较。如相等则验证通过。该过程也可以在证书签发时进行。

8.1.3 证书的签发

证书签发由证书签发系统完成，包括根证书、CA 证书以及用户证书的签发。

a) 根证书和 CA 证书的签发：根证书是一张自签名证书，使用证书中的公钥即可验证证书的签名。在系统初始化时，首先要签发一张根证书。下级 CA 的数字证书由上级 CA 签发。在证书中需要在扩展域标识该证书可以用来签发证书。

b) 用户证书的签发：认证系统签发用户证书时，首先根据用户的申请信息，以及审核信息确认是否为该用户签发证书，当确认可以签发后，向密钥管理中心申请一对加密密钥，再根据申请信息为用户签发两张数字证书并将两张数字证书发布到目录服务器上，然后将数字证书以及加密证书的私钥回传给用户。

8.1.4 证书的下载

a) 根证书、CA 证书的下载：根证书、CA 证书可以由用户通过证书/证书撤销列表存储与发布系统下载，也可以与用户证书一起下载。

b) 用户证书的下载：用户或其代理者进行证书的下载时，首先向 RA 提供确认信息。通过确认后，将签发好的用户证书和加密证书的私钥，下载到用户的证书载体中。

8.1.5 证书的注销

证书注销由证书/证书撤销列表生成与签发系统完成，分为两种情况：

a) 强制注销：证书认证系统的管理人员可以在策略规定的范围内强制注销证书。

b) 用户申请注销：当用户因某种原因不再或不能使用证书时，可以通过 RA 申请注销证书。

证书注销的过程与证书的申请过程相同。

8.1.6 证书撤销列表的发布

a) 发布的时间策略：可以采取实时发布和定时发布两种策略。实时发布是指签发系统接到注销请求后，立刻根据请求信息签发注销列表；定时发布是指签发系统接到注销请求信息后不立刻签发注销列表，而是按照系统的设定，在确定的时间里签发注销列表；

b) 发布的形式：可以采用完全的注销列表、增量证书撤销列表以及证书分布点技术发布证书撤销列表。

8.1.7 证书的更新

证书的更新包括根证书、CA 证书的更新和用户证书的更新：

a) 根证书和 CA 证书更新：

根证书和 CA 证书密钥更新根据证书密钥更新的策略进行。

证书密钥更新时，证书认证系统需要签发三个新证书：

1) 新私钥签名的包含新公钥的证书；

2) 新私钥签名的包含旧公钥的证书；

3) 旧私钥签名的包含新公钥的证书。

在过渡期中，系统中存在着的四个证书，保证所有实体能够在各种情况下对所接到的证书进行验证。过渡期结束，只保留新私钥签名的包含新公钥的证书。

b) 用户证书更新：

用户证书的更新包括签名证书的更新和加密证书的更新。

用户证书更新应向RA提出申请。证书/证书撤销列表生成与签发系统接到证书更新申请后，首先将旧的证书作废，如果是加密证书更新则向密钥管理中心申请新的加密密钥对，然后签发新的证书，将新证书传递给用户并发布到证书/证书撤销列表存储与发布系统中。

8.2 证书验证协议

8.2.1 概述

用户在使用数字证书进行加密和验证数字签名时，必须首先验证证书的有效性，验证证书的有效性包括三个方面的内容：

a) 用CA的证书验证用户证书中的签名，确认该证书是该CA签发的，并且证书的内容没有被篡改；

b) 检验证书的有效期，确认该证书在有效期之内；

c) 查询CRL，确认该证书没有被注销。

8.2.2 认证路径

在进行证书验证时，需要根据证书的签发者查询签发者证书，并验证其有效性，直到找到一个预先确定的可信任的CA证书，在这个过程中，形成了一个包含多个CA证书的证书列表，这个列表就是证书的认证路径。

证书认证路径的获取可以在用户申请证书之前从CA下载，也可以在需要时实时分别从不同CA下载。

有关认证路径的具体处理过程，参见国家相关标准。

8.2.3 证书状态查询

证书状态查询为用户和应用提供查询证书状态的查询服务。

a) CRL的获取：用户或应用系统可通过证书中的CRL地址标识下载；

b) CRL验证：验证时，首先检查CRL文件是否在有效期间内，否则重新下载；然后验证CRL的签名以确认其正确性；最后检查CRL文件中是否包含所需要验证的证书的序列号，如果包含则说明该证书已经被注销；

c) dCRL验证：dCRL中包含了最新注销的证书信息，dCRL需要与某一个基本的CRL一起才能验证。验证的方法与CRL相同。

8.2.4 在线证书状态查询协议(OCSP)

使用在线证书状态查询需要客户端与OCSP服务器保持实时的连接，证书中包含OCSP服务器的地址，通过这个地址，可以使用在线证书状态查询服务。

具体的查询过程，参见国家相关标准。

8.2.5 简明在线证书状态查询协议(SOCSP)

简明在线证书状态查询服务为用户和应用系统提供快速在线状态查询服务。

具体的查询协议，参见国家相关标准。

8.3 安全通信协议

证书认证系统各子系统之间需要采用安全通信协议以保证通信安全。

有关安全通信协议的详细内容可参见附录B。

9 证书认证中心建设

9.1 系统

9.1.1 功能要求

CA提供的服务功能主要有：

a) 提供各种证书在其生命周期中的管理服务；

b) 提供RA的多种建设方式，RA可以全部托管在CA系统，也可以部分托管在CA，部分建在远端；

c) 提供人工审核或自动审核两种审核模式；

d) 支持多级 CA 认证；

e) 提供证书查询、证书状态查询、证书撤销列表下载、目录服务等功能。

9.1.2 性能要求

CA 系统的性能应满足如下要求：

a) 系统对用户接口采用标准的 HTTP、HTTPS 和 LDAP 协议，确保各种用户都能够使用本系统服务；

b) 系统各模块的状态信息保存在配置文件和数据库内部，保证系统的部署方便性和配置方便性，当系统需改变配置时无需中断系统的服务；

c) 各模块的功能可以通过配置文件进行控制，系统可以根据不同的需求进行设置；

d) 系统某一功能模块可有多个实例，并且多个实例可运行在一台或多台计算机上；

e) 系统应有冗余设计，保证系统的不间断运行。

9.1.3 管理员配置要求

在 CA 应设置下列管理和操作人员：

——超级管理员；

——审计管理员；

——业务管理员；

——业务操作员。

"超级管理员"负责 CA 系统的策略设置，设置各子系统的业务管理员并对其管理的业务范围进行授权。

"业务管理员"负责 CA 系统的某个子系统的业务管理，设置本子系统的业务操作员并对其操作的权限进行授权。

"业务操作员"按其权限进行具体的业务操作。

"审计管理员"负责对涉及系统安全的事件和各类管理和操作人员的行为进行审计和监督。

上述各类人员使用证书进行登录，其中"超级管理员"和"审计管理员"的证书应在 CA 系统进行初始化时同时产生。

另外，CA 应设置安全管理员，全面负责系统的安全工作。

9.1.4 网络划分

CA 系统的计算机网络需要合理分段，原则上要求整个网络应划分为四部分：

a) 公共部分：为 CA 用户所在的网络，所有用户将通过该网络访问 CA；

b) 服务部分：为外部用户提供域名解析功能，并负责内部系统对外邮件的收发功能；包括系统的各种 Web 服务器和从目录服务器，是外部用户访问内部功能的接口，为用户提供访问界面；

c) 管理部分：仅供 CA 的工作人员使用的网络；

d) 核心部分：包括各种核心应用、数据库和密码设备等在内的实现系统功能的安全网络。

当 RA 采用客户机/服务器(C/S)模式时，应该按照上述方式划分网络；当 RA 采取浏览器/服务器(B/S)模式时，可将服务与管理网络放在同一网段。网络结构示意图参见附录 D《证书认证系统网络结构图》。

9.1.5 初始化要求

CA 的初始化过程必须完成下列工作：

a) 产生本 CA 的机构密钥并安全备份；

b) 若本 CA 为根 CA，则使用根 CA 的签名密钥进行自签名；若本 CA 从属于某一根 CA，则将产生的签名公钥提交根 CA 签发本 CA 的证书；

c) 由 CA 签发 CA 服务器证书；

d) 由 CA 签发 RA 服务器证书(可选);

e) 由 CA 签发超级管理员和审计管理员证书;

f) 由 CA 签发其他管理员和操作员证书。

9.2 安全

9.2.1 概述

CA 系统的安全包括系统安全、通信安全、密钥安全、证书管理安全、安全审计、物理安全、人员安全等各方面的安全。

9.2.2 系统安全

系统安全的主要目标是保障网络、主机系统、应用系统及数据库运行的安全。应采取防火墙、病毒防治、漏洞扫描、入侵监测、数据备份、灾难恢复等安全防护措施。

9.2.3 通信安全

通信安全的主要目标是保障 CA 系统各子系统之间、CA 与 KMC 之间、CA 与 RA 之间的安全通信,应采取通信加密、安全通信协议等安全措施。

9.2.4 密钥安全

9.2.4.1 概述

密钥安全的主要目标是保障 CA 系统中所使用的密钥,在其生成、存储、使用、更新、废除、归档、销毁、备份和恢复整个生命周期中的安全。应采取硬件密码设备、密钥管理安全协议、密钥存取访问控制、密钥管理操作审计等多种安全措施。

9.2.4.2 基本要求

密钥安全的基本要求是:

a) 密钥的生成和使用必须在硬件密码设备中完成;

b) 密钥的生成和使用必须有安全可靠的管理机制;

c) 存在于硬件密码设备之外的所有密钥必须加密;

d) 密钥必须有安全可靠的备份恢复机制;

e) 对密码设备操作必须由多个操作员实施。

9.2.4.3 根 CA 密钥

根 CA 密钥的安全性除了满足基本要求外,还应满足下列要求:

a) 根 CA 密钥的产生:

CA 系统的根密钥由硬件密码设备生成并存放在该密码设备中,应采用密钥分割或秘密共享机制进行备份,保存分割后的根密钥的人员称为分管者。

生成根 CA 密钥时,应先选定分管者,数量可以限定为 3 个或 5 个。选定的分管者应分别用自己输入的口令保护分管的密钥,分管的密钥应存放在智能 IC 卡或智能密码钥匙中。智能 IC 卡或智能密码钥匙也应备份,并安全存放。

根 CA 密钥的产生过程必须进行记录。

b) 根 CA 密钥的恢复:

恢复根 CA 密钥时,要有满足根 CA 密钥恢复所必需的分管者人数。各个分管者输入各自的口令和分管的密钥成份在密码设备中恢复。

c) 根 CA 密钥的更新:

根 CA 密钥的更新,需重新生成根 CA 密钥,其过程同根 CA 密钥的生成。

d) 根 CA 密钥的废除:

根 CA 密钥的废除应与根 CA 密钥的更新同步。

e) 根 CA 密钥的销毁:

根 CA 密钥的销毁应与备份的根 CA 密钥一同销毁。由密码主管部门授权的机构实施。

9.2.4.4 非根 CA 密钥

非根 CA 密钥的安全性要求与根 CA 密钥的安全性要求一致。

9.2.4.5 管理员证书密钥

管理员包括超级管理员、审计管理员、业务管理员和业务操作员等。管理员证书密钥应由证书载体来产生和存储。

管理员证书密钥的安全性应满足下列要求：

a) 管理员证书密钥的产生和使用必须在证书载体中完成；

b) 密钥的生成和使用必须有安全可靠的管理机制；

c) 管理员的口令长度为 8 个字节以上；

d) 管理员的账号要和普通用户账号严格分类管理。

9.2.5 证书管理安全

证书的管理安全应满足下列要求：

a) 验证证书申请者的身份；

b) 防止非法签发和越权签发证书，通过审批的证书申请必须提交给 CA，由 CA 签发与申请者身份相符的证书；

c) 保证证书管理的可审计性，对于证书的任何处理都应作日志记录。通过对日志文件的分析，可以对证书事件进行审计和跟踪。

9.2.6 安全审计

9.2.6.1 概述

CA 系统在运行过程中涉及大量功能模块之间的相互调用，以及各种管理员的操作，对这些调用和操作需要以日志的形式进行记载，以便用于系统错误分析、风险分析和安全审计等工作。

9.2.6.2 功能模块调用日志

系统内的各功能模块在运行过程中会调用其他功能模块或被其他功能模块所调用，对于这些相互之间的功能调用，各模块应该记录如下数据：

a) 调用请求的接收时间；

b) 调用请求来自的网络地址；

c) 调用请求发起者的身份；

d) 调用请求的内容；

e) 调用请求的处理过程；

f) 处理结果等。

9.2.6.3 CA 系统管理员审计

CA 系统管理员的下列操作应被记录：

a) 根 CA 证书加载；

b) CA 证书加载；

c) 证书撤销列表加载；

d) 证书撤销列表更新等。

e) 系统配置；

f) 权限分配。

9.2.6.4 CA 业务操作员审计

CA 业务操作员的下列操作应被记录：

a) 证书请求批准；

b) 证书请求拒绝；

c) 证书请求分配；

d) 证书注销。

9.2.6.5 RA 业务操作员审计

RA 业务操作员的下列操作应被记录：

a) 证书请求批准；

b) 证书请求拒绝；

c) 证书请求分配；

d) 证书注销。

9.3 数据备份

数据备份的目的是确保 CA 的关键业务数据在发生灾难性破坏时，系统能够及时和尽可能完整地恢复被破坏的数据。应选择适当的存储备份系统对重要数据进行备份。

不同的应用环境可以有不同的备份方案，但应满足以下基本要求：

a) 备份要在不中断数据库使用的前提下实施；

b) 备份方案应符合国家有关信息数据备份的标准要求；

c) 备份方案应提供人工和自动备份功能；

d) 备份方案应提供实时和定期备份功能；

e) 备份方案应提供增量备份功能；

f) 备份方案应提供日志记录功能；

g) 备份应提供归档检索与恢复功能。

9.4 可靠性

9.4.1 概述

CA 必须提供 7×24 h 服务，对影响系统可靠性的主要因素如网络故障、主机故障、数据库故障和电源故障等应采取冗余配置等措施。

9.4.2 网络链路冗余

为保证 CA 的服务，CA 网络对外接口应根据具体情况，可有两条物理上独立的链路，同时考虑交换机、路由器、防火墙的冗余配置。

9.4.3 主机冗余

CA 系统中与关键业务相关的主机、在服务网段和核心网段中的服务器应采用双机热备份或双机备份措施。

9.4.4 数据库冗余

CA 系统的数据库应采用磁盘阵列、磁盘镜像等措施，具备容错和备份能力。

9.4.5 电源冗余

CA 系统应采用高可靠的电源解决方案，并应采用 UPS 为系统提供不间断电源。

9.5 物理安全

9.5.1 物理环境建设

CA 的建筑物及机房建设应按照国家密码管理相关政策要求，并按照下列标准实施：

a) GB/T 9361—1988；

b) GB/T 2887—2000；

c) SJ/T 10796—1996；

d) GB 50174—2008。

9.5.2 对 CA 的分层访问

9.5.2.1 概述

CA 系统按功能分为四个区域，由外到里分别是：公共区、服务区、管理区和核心区，各区的功能及设备配置参见附录 D。

9.5.2.2 公共区

入口之外的区域为公共区。

9.5.2.3 服务区

所有进入此区的人员使用身份识别卡刷卡进入。该区的每扇窗户都应安装玻璃破碎报警器。

9.5.2.4 管理区

所有进入此区人员需要同时使用身份识别卡和人体特征鉴别才可以进入，人员进出管理区要有日志记录。所有的房间不应安装窗户，所有的墙体应采用高强度防护墙。

9.5.2.5 核心区

所有进入此区人员需要同时使用身份识别卡和人体特征鉴别才可以进入，人员进出该区要有日志记录。

核心区应为屏蔽机房，应加装高强度的钢制防盗门。所有进出屏蔽室的线路都要采取防电磁泄漏措施。屏蔽效果应符合国家密码管理相关政策要求并达到国家相关标准要求。

9.5.2.6 安全监控和配电消防

CA 应设置安全监控室、系统监控室、配电室和消防器材室。

安全监控室是安全管理人员值班的地方，可对整个 CA 的进出人员实行监控，处理日常的安全事件。只有安全管理人员同时使用身份识别卡和人体特征鉴别才可以进入，刷卡离开。

系统监控室是网络管理人员工作的地方。需要同时使用身份识别卡和人体特征鉴别才可以进入，刷卡离开。

配电室是放置所有供电设备的房间，只有相应的授权人员同时使用身份识别卡和人体特征鉴别才可以进入，刷卡离开。

消防器材室是存放消防设备的房间，建议使用身份识别卡进入消防器材室。

9.5.3 门禁和物理侵入报警系统

CA 应设置门禁和物理侵入报警系统。

门禁系统控制各层门的进出。工作人员都需使用身份识别卡或结合人体特征鉴别才能进出，并且进出每一道门都应有时间记录和相关信息提示。

任何非法的闯入、非正常手段的开门、以及授权人刷卡离开后房内还有非授权的滞留人员，都应触发报警系统。报警系统应明确地指出报警部位。

门禁和物理侵入报警系统应自备有 UPS，并应提供至少 8 h 的供电。

与门禁和物理侵入报警系统配合使用的还应有录像监控系统。对监控区域进行 24 h 不间断的录像。所有的录像资料要根据需要保留一段时间，以备查询。

9.6 人事管理制度

人事管理制度包括人员的可信度鉴别、岗位设置等。

CA 应制定可信人员策略并据此进行人员的可信度鉴别和聘用。可信人员必须接受并通过广泛的背景调查，才能证明他们有能力进行那些关键操作所必需的信任级别。

CA 对人员的教育水平、从业经历、信用情况等方面进行调查，来评估人员的可信度。进行可信人员背景调查必须遵循国家的有关法律、法规和政策。

10 密钥管理中心建设

10.1 建设原则

密钥管理中心的工程建设按照与 CA 统一规划、有机结合、独立设置、分别管理的原则建设。

10.2 系统

10.2.1 功能要求

密钥管理中心应提供下列服务功能：

a) 为 CA 提供密钥生成服务；

b) 为司法机关提供密钥恢复服务；

c) 为用户提供密钥更新、密钥恢复、密钥撤销服务。

10.2.2 性能要求

密钥管理中心的性能应满足如下要求：

a) 密钥的保存期应大于 10 年；

b) 系统应支持多并发服务请求；

c) 系统各模块的状态信息保存在配置文件和数据库内部，保证系统的部署方便性和配置方便性，当系统需改变配置时无需中断系统的服务；

d) 各模块的功能可以通过配置文件进行控制，系统可以根据不同的需求进行设置；

e) 系统应有冗余设计，保证系统的不间断运行。

10.2.3 管理员配置要求

在 KMC 应设置下列管理和操作人员：

——超级管理员

——审计管理员

——业务管理员

——业务操作员

“超级管理员”负责 KMC 系统的策略设置，设置各子系统的业务管理员并对其管理的业务范围进行授权。

“业务管理员”负责 KMC 系统的某个子系统的业务管理，设置本子系统的业务操作员并对其操作的权限进行授权

“业务操作员”按其权限进行具体的业务操作。

“审计管理员”负责对涉及系统安全的事件和各类管理和操作人员的行为进行审计和监督。

上述各类人员使用证书进行登录，其中“超级管理员”和“审计管理员”的证书应在 KMC 系统进行初始化时同时产生。

另外，KMC 应设置安全管理员，全面负责系统的安全工作。

10.2.4 初始化要求

KMC 的初始化过程必须完成下列工作：

a) 生成 KMC 的机构密钥并安全备份；

b) 由授权的 CA 签发 KMC 服务器证书；

c) 由授权的 CA 签发超级管理员和审计管理员证书；

d) 由授权的 CA 签发业务管理员和业务操作员证书。

10.3 安全

KMC 的安全参照本标准 9.2 的要求进行。

10.4 数据备份

KMC 的数据备份参照本标准 9.3 的要求进行。

10.5 可靠性

KMC 的可靠性参照本标准 9.4 的要求进行。

10.6 物理安全

KMC 的物理安全参照本标准 9.5.2.5 的要求进行。

10.7 人事管理制度

KMC 的人事管理制度参照本标准 9.6 的要求进行。

11 证书认证中心运行管理要求

11.1 人员管理要求

为防止非授权人员操作CA系统，在每一个操作终端上应设有操作员身份鉴别系统，对系统的所有操作都要对有关操作员进行身份鉴别和权限控制。

CA系统的每个操作人员配置有标明个人身份与相关资料的证书载体，证书载体具有口令保护机制，以保证私钥和应用的安全。

人员管理的主要内容是：增加操作员、注销操作员、设置操作员权限、修改操作员权限。

操作员信息包括：操作员编号、操作员姓名、操作员部门、操作员权限。

超级管理员由系统初始化时产生，主要职责是设置业务管理员并进行管理。其权限为：

a) 增加业务管理员；

b) 注销业务管理员；

c) 设置业务管理员权限；

d) 修改业务管理员权限。

业务管理员由超级管理员授权，主要职责是设置业务操作员并进行管理。其权限为：

a) 增加业务操作员；

b) 注销业务操作员；

c) 设置业务操作员权限；

d) 修改业务操作员权限。

业务操作员由业务管理员授权，主要职责是对业务系统进行各种操作。

审计管理员由系统初始化时产生，主要职责是负责对涉及系统安全的事件和各类管理和操作人员的行为进行审计和监督。其权限为：

a) 证据访问操作；

b) 日志访问操作。

管理员和操作员登录CA系统以及在CA中的所有操作都采用基于证书的身份鉴别。当此类人员离职或是被撤职时，应及时注销其证书。

11.2 CA业务运行管理要求

11.2.1 概述

CA应制定业务运行管理规范来指导CA日常业务开展。业务运行管理规范通常应包括CA管理制度、信息系统安全操作与维护以及客户服务等。

11.2.2 CA管理制度

CA管理制度包括CA运行场所进出管理制度、客户信息保密制度、CA工作人员管理制度、机房安全管理制度等，应按国家有关标准执行。

11.2.3 安全操作与维护规范

11.2.3.1 系统管理

系统管理的操作与维护规范应包括以下内容：

a) 对CA系统进行任何操作之前，应充分考虑并预计操作之后的结果，每次操作都必须记录；

b) 改变系统的配置，应制订实施计划和相关文档说明，经上级主管批准后才能进行操作，操作时应有双人在场；

c) 系统出现故障时，应由系统管理人员检查处理，其他人员未经批准不得处理；

d) 未经批准不得在服务器上安装任何软件和硬件；

e) 未经批准不得删除服务器上的任何文件。

11.2.3.2 数据备份

数据备份的操作与维护规范应包括以下内容：

a) 系统升级后，应立即进行全备份；

b) 对数据变化量大的服务器，应每天做一次增量备份，每周做一次全备份；

c) 对数据变化量少的服务器，可每周做一次备份；

d) 对重要数据应准备两套备份，其中异地存放一套；

e) 对数据库的备份应单独进行；

f) 对重要的目录应单独进行备份；

g) 手工进行的备份，应在介质上标明备份的服务器及路径；

h) 自动进行的备份，应将备份介质有效区分；

i) 选择的备份介质应能保证数据的长期可靠，否则应定期更新。

11.2.3.3 口令管理

口令管理规范应包括以下内容：

a) 口令长度应为8个字节以上，应是字母、数字和特殊字符组成的混合体，口令不得采用有特殊意义的(如姓名、生日、电话号码等)数字和词组；

b) 应规定口令的使用期限并定期更换；

c) 口令应妥善保管，防止泄漏；

d) 通过网络传输的口令必须保护；

e) 应检查网络设备、主机和应用程序中是否设置有缺省口令的缺省用户名，找出并禁止。

11.2.3.4 应急处理

CA应制订应急处理预案，当出现重大故障或灾难性事故时，应启动预定的应急处理方案进行处理。

应急处理预案应根据事件的严重程度、紧急程度和事件类别，分别规范告警、报告、保护、处置、善后、总结等处理流程和处置措施。

系统恢复正常运行后，应对应急处理过程进行总结，总结中应详细记录事件起因、处理过程、经验教训、改进建议等。

应针对应急事件处理中暴露的问题，不断完善和修改应急处理预案。

11.2.4 客户服务规范

CA应对客户提供全面、及时、有效的服务，保证客户在证书使用过程中出现的任何问题都能及时得到响应和解决。

服务的过程应作记录。

11.3 密钥分管要求

CA和KMC的根密钥需要用密钥分割或秘密共享机制分割备份出来，分别交予分管者保管。恢复时，到场的分管者的人数应满足恢复所需的人数。

分管者的选择条件如下：

a) 分管者应符合可信人员策略规定的条件；

b) 符合下列条件之一者，不能成为分管者：

1) 本证书认证系统的超级管理员；

2) 本证书认证系统的业务管理员；

3) 本证书认证系统的业务操作员；

4) 本证书认证系统的系统维护人员。

11.4 安全管理要求

安全管理员的职责主要包括：

a) 制定 CA 的安全策略；
b) 指导 CA 的安全管理；
c) 设计和指导 CA 的安全策略实施；
d) 对 CA 的安全管理进行定期的检查和评估；
e) 对安全策略和执行程序的日常维持；
f) 定期对相关人员开展安全教育。

安全管理员对安全的三个关键领域负有全面的责任，即：

a) 开发与执行安全策略；
b) 维护与完善安全策略；
c) 保持与安全审计的一致性。

安全管理员有责任来定义和委托 CA 的特定个人或部门的安全职责。

11.5 安全审计要求

审计管理员应定期对 CA 进行安全审计，包括：

a) 人员审计：CA 的人员必须是可信任的；必须理解安全策略和安全操作程序；
b) 物理安全审计：物理安全防护措施是否完善；安全物品的管理是否符合 CA 的安全管理规定；
c) 通信安全审计：CA 的所有安全通信设备的使用是否符合 CA 的安全管理规定；
d) 操作安全审计：CA 所有的人员的操作记录必须完整保存，并且所有操作必须符合 CA 的安全管理规定；
e) 系统安全审计：检查 CA 的操作系统、数据库系统、入侵检测系统、漏洞扫描系统、防病毒系统、防火墙系统、CA 系统等的日志记录，以确定系统是否异常。

11.6 文档配备要求

11.6.1 概述

CA 应配备相关的文档用于指导 CA 的建设、运行、服务、应急和日常管理。可分为技术实现、物理建设、人事管理、运行管理以及审计与评估五类。

11.6.2 技术实现类

技术实现类主要包括 CA 系统设计、CA 系统安全、CA 系统安装与配置手册、CA 系统安全目标、CA 系统用户手册五类文档，技术实现类文档主要描述内容如下：

a) CA 系统设计：描述 CA 系统的逻辑结构、网络结构、数据通信设计、密钥管理、业务处理流程以及系统的软硬件配置等。
b) CA 系统安全：描述 CA 系统通过采用防火墙、入侵检测、漏洞扫描、病毒防治、访问控制、安全配置等措施，保证 CA 的安全性。同时，从数据通讯、密钥管理、证书管理、安全审计、物理安全等各个方面阐述 CA 安全措施的实现。
c) CA 系统安装与配置手册：介绍 CA 系统的安装与配置。
d) CA 系统安全目标：描述 CA 系统对国家相关安全标准的满足情况。
e) CA 系统用户手册：描述用户对 CA 系统使用和操作的技术手册。

11.6.3 物理建设类

物理建设类主要包括物理场地安全手册、物理场地安全管理规定两类文档，物理建设类文档主要描述内容如下：

a) 物理场地安全手册：描述物理场地的安全的要求及实现等；
b) 物理场地安全管理规定：描述人员进出 CA 各个区域的权限、来访者的接待和管理、门禁系统的使用、监控报警系统的操作使用等管理规定。

11.6.4 人事管理类

人事管理类文档主要包括可信人员策略、可信人员职位划分原则与鉴别两类文档，人事管理类文档

主要描述内容如下：

a) 可信人员策略：描述可信人员策略及其如何进行可信人员调查；

b) 可信人员职位划分原则与鉴别：描述可信人员职位划分原则，可信人员鉴别和背景调查及分析等。

11.6.5 运行管理类

运行管理类文档主要包括账号管理、CA 管理规范、认证业务声明、操作手册、安全应急预案、客户报务规范六类文档，运行管理类文档主要描述内容如下：

a) 账号管理：描述账号的处理和管理；

b) CA 管理规范：描述 CA 的操作与安全维护管理的规定；

c) 认证业务声明：对外公布的证书认证业务服务声明；

d) 操作手册：描述认证业务流程；

e) 安全应急预案：描述 CA 电力系统、消防系统、业务系统、人员变动、安全等方面出现事故时的应急处理流程和措施；

f) 客户服务规范：是由 CA 制定出的系列客户服务文档，包括客户法律协议、隐私保护政策、客户保障计划等。

11.6.6 审计与评估类

审计与评估类文档主要包括 CA 安全与审计规范、安全审核与评估规范两类文档，审计与评估类文档主要描述内容如下：

a) CA 安全与审计规范：规定了 CA 运行系统的审核方法；

b) 安全审核与评估规范：规定了 CA 运行系统的审核范围和评价标准。

12 密钥管理中心运行管理要求

12.1 人员管理要求

按本标准 11.1 的要求执行。

12.2 运行管理要求

按本标准 11.2 的要求执行。

12.3 密钥分管要求

按本标准 11.3 的要求执行。

12.4 安全管理要求

按本标准 11.4 的要求执行。

12.5 安全审计要求

按本标准 11.5 的要求执行。

12.6 文档配备要求

按本标准 11.6 的要求执行。

13 检测

13.1 概述

证书认证系统建成后，应按照本章要求对证书认证系统进行功能、性能以及安全性方面的检测。测试部门可根据本标准的要求及被测系统的具体情况，制订测试大纲，进行测试。

13.2 系统初始化

系统应能按照本标准 9.1.5 和 10.2.4 的要求正确进行初始化。

13.3 用户注册管理系统

13.3.1 用户证书申请信息的录入功能

系统应能对认证系统定义的各类证书所要求的信息正确地录入，对于证书本身所需信息之外的用户信息也能够正确地录入。

系统应能批量读取按照系统规定格式存储在外部介质中的或通过网络传输的用户证书申请信息。

13.3.2 用户信息审核功能

系统应能读取需要进行审核的证书申请信息，与可以证明用户真实身份的信息进行比较和审核。应能将审核通过的信息正确提交给签发系统并反馈给证书申请者。不能通过审核的信息应退回录入者并说明理由。

13.3.3 用户证书下载功能

系统应能根据认证系统的规定和用户的选择，正确地下载证书，并且将证书和私钥信息安全地写入证书载体中。

13.3.4 系统的多级审核功能

系统应能正确地将信息提交给上一级注册管理系统，并能够根据系统的策略，对不同种类的用户采用不同的审核模式；

13.3.5 用户信息归档和恢复功能

系统应能自动或人工进行用户信息的归档，并且能够根据归档的用户信息将系统的数据恢复到某一时刻的状态。

13.3.6 日志处理功能

系统应能准确清晰地记录日志，应能提供完善的查询、归档和防篡改能力。

日志中应记录包括事件发生的时间、事件的请求者和执行者以及系统和相关者的签名等信息。

13.4 证书/证书撤销列表生成与签发系统

13.4.1 多层结构支持功能

系统应能将所下载的根证书和下级 CA 证书导入到证书存储区中，并能正确识别出根证书和下级 CA 证书，建立正确的认证路径。

13.4.2 CRL 签发功能

系统应能根据 CRL 签发策略正确签发 CRL 文件，并且能够通过证书中的 CRL 分布点地址，进行 CRL 的查询和下载。

13.4.3 证书模板功能

系统应能根据用户的要求，使用证书模板对所签发的证书类型及内容进行灵活的定义。

13.4.4 CA 证书签发功能

系统应能利用签名密钥签发证书。

13.4.5 CA 证书更新功能

系统应能正确完成 CA 证书的更新，在证书认证系统的发布系统中，生成新的 CA 与旧的 CA 证书的认证证书链，供不同情况下为用户提供证书验证服务。

13.4.6 证书归档和恢复功能

系统应能自动或人工进行证书的归档，并且能够根据归档证书将系统的数据恢复到某一时刻的状态，实现系统的恢复功能。

13.4.7 用户证书更新功能

系统应能为已经注册的用户重新签发证书，并将证书和加密证书的私钥安全地传递到用户的证书载体中。在证书载体中，不仅存放新的证书和私钥，还要存放旧的证书和私钥。

13.4.8 并发处理能力

系统应能实现所声明的并发处理能力。

13.4.9 日志处理功能

系统应能准确清晰地记录日志，应能提供完善的查询、归档和防篡改能力。

日志中应记录包括事件发生的时间、事件的请求者和执行者以及系统和相关者的签名等信息。

13.4.10 管理员配置功能

超级管理员和审计管理员必须是平级的关系，并且在系统初始化时同时生成。审计管理员独立于超级管理员，不能由超级管理员来进行授权。

超级管理员能够添加、删除业务管理员，并能够为其分配权限、申请和制作证书。

业务管理员能够添加、删除业务操作员，并能够为其分配权限、申请和制作证书。

审计管理员能够审计超级管理员、业务管理员和业务操作员的全部操作。

13.5 证书/证书撤销列表存储与发布系统

13.5.1 数据库备份和恢复功能

系统应能自动或人工进行数据库信息的备份，并且能够根据备份信息将系统的数据恢复到某一时刻的状态。

13.5.2 目录服务器的目录信息树定义功能

系统应能进行目录服务系统的目录信息树的结构定义，能够根据认证系统的具体要求，灵活地定制信息树的结构。

13.5.3 目录服务器管理功能

系统应能正确地将证书信息以及 CRL 信息写入目录服务器中；能够根据一定的查询条件，查询有关用户和证书以及 CRL 的信息。

13.5.4 目录服务器的主从映射功能

系统应能将信息自动地由主目录服务器实时映射到从目录服务器中，可支持一主多从的目录服务器结构。

13.5.5 目录服务器备份和恢复功能

系统应能自动或人工进行目录服务器信息的备份，并且能够根据备份信息将系统的数据恢复到某一时刻的状态。

13.5.6 数据库系统和目录服务器一致性检验功能

系统应能对在数据库系统和目录服务器系统中的信息进行一致性检查，当出现信息不一致的情况时，管理人员能够根据策略要求进行修改。

13.6 证书状态查询系统

13.6.1 CRL 查询功能

系统应能提供 CRL 查询功能，使用户或应用系统利用证书中标识的 CRL 的地址，查询并下载。

13.6.2 OCSP 功能。

系统应能提供在线证书状态查询功能，查询的返回信息带有 OCSP 服务器的数字签名，并能正确返回所查询的证书状态。

13.6.3 SOCSP 功能

系统应能提供简易在线证书状态查询功能，查询的返回信息带有 SOCSP 服务器的数字签名，并能正确返回所查询的证书状态。

此功能为可选项。

13.7 安全审计系统

系统应能分别查询各子系统中的日志记录，也可以通过查询证书/证书撤销列表存储与发布系统的数据库，进行审计。

系统应能对证书认证系统中的有关事件、管理员的操作行为等进行审计，能够根据时间、事件、人员等条件进行统计。

13.8 密钥管理中心检测

13.8.1 密钥生成与分发

系统能够支持预生成和实时生成密钥对两种方式：

——系统能够根据配置参数，在确定的时间，正确地批量生成密钥对；

——系统能够在 CA 提出密钥申请时，实时地分发密钥对。

13.8.2 密钥状态转移

系统密钥状态转移应满足：

——在系统生成密钥对后，所有生成的密钥对都在备用库中；

——当密钥分发给用户后，密钥在在用库中；

——当用户的证书注销后，证书中对应的密钥在历史库中；

——密钥库中的密钥必须加密存放。

13.8.3 密钥信息归档和密钥恢复

系统密钥信息归档和密钥恢复应满足：

——系统能够提供密钥信息归档功能；

——能够自动/人工进行密钥信息的归档；

——当且仅当符合操作规程时可以实现密钥恢复功能。

13.8.4 司法取证密钥恢复

系统能够在符合相关管理规定的条件下，提供司法取证密钥恢复服务。恢复的密钥不能以明文的形式出现在载体之外，加密该密钥的密钥也不能以明文的形式出现在载体之外。

13.8.5 管理功能

人员管理应符合人员管理制度；运行管理应符合运行管理流程，应具备应急处理能力。

13.9 系统安全性检测

13.9.1 密钥管理安全检测

密钥管理安全应满足：

a) 密钥的生成必须由国家密码主管部门认可的密码设备生成；

b) 系统中使用的密钥必须存储在硬件密码设备中；

c) 存储在密码设备之外的密钥必须加密；

d) 密钥必须加密传输。

13.9.2 系统的访问控制检测

系统的访问控制应满足：

a) 非授权人员无法访问系统；

b) 非授权的 IP 地址的主机无法访问被保护的主机；

c) 管理和操作人员不能进行非授权的操作。

13.9.3 系统的审计日志检测

系统的审计日志应满足：

a) 操作事件的日志必须具有数字签名；

b) 日志不可篡改。

13.9.4 系统的数据安全检测

系统的数据安全应满足：

a) 保护口令等敏感信息必须加密存储；

b) 各子系统之间通信必须进行身份鉴别和加密传输。

13.10 其他安全产品和系统

其他安全产品和系统包括防火墙、入侵检测、漏洞扫描、病毒防治等安全产品以及机房、门禁、供电、消防、电磁泄漏、数据备份、灾难恢复等物理安全措施，检验内容及其标准如下：

a) 采用符合国家相关标准和规范、并取得相应资质的产品；

b) 根据系统的具体情况对产品进行了正确的配置和设置。

附 录 A
（资料性附录）
KMC 与 CA 之间的消息格式

A.1 概述

KMC 为 CA 提供用户加密密钥对。KMC 在接收到来自 CA 的请求后，首先检查请求的合法性与正确性，然后根据 CA 的请求进行相应的处理，并将结果返回给 CA。

KMC 为 CA 提供服务的完整过程包括请求、响应、回执，以及异常情况的处理。

A.1.1 请求

请求指来自 CA 的请求，包含 CA 请求的类型、性质以及特性数据等，该请求将被发送到 KMC 并得到服务。服务请求包括如下内容：

——协议版本；

——服务请求标识符；

——CA 证书标识符；

——扩展的请求信息；

——请求信息的签名。

A.1.2 响应

响应指 KMC 对来自 CA 请求的处理响应。KMC 的响应包括如下内容：

——协议版本；

——响应标识符；

——KMC 证书标识符；

——响应信息；

——响应信息的签名。

A.1.3 回执

回执指 CA 在接受到 KMC 的响应数据后，应答 KMC 发送的回执信息。回执包括如下内容：

——协议版本；

——回执标识符；

——CA 证书标识符；

——回执信息；

——回执信息的签名。

A.1.4 异常情况

当 KMC 和 CA 任何一方发生错误时，均需要向对方发送错误信息。错误可以是下列几类：

——验证请求失败：KMC 验证来自 CA 证书或 CA 请求数据失败，CA 收到后应重新进行申请；

——内部处理失败：KMC 处理 CA 的请求过程中发生内部错误，通知 CA 该请求处理失败，需要重新申请；

——验证响应失败：CA 验证 KMC 证书或者来自 KMC 的响应数据失败；

——验证回执失败：KMC 验证来自 CA 的回执信息失败，通知 CA 需要重新处理。

A.2 协议

A.2.1 约定

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。

引用的其他术语还有：Extensions，CertificateSerialNumber，SubjectPublicKeyInfo，Name，AlgorithmIdentifier，CRLReason。

A.2.2 请求

A.2.2.1 请求数据格式

CA 请求的基本格式如下：

```
CARequest ::=SEQUENCE {
ksRequest            TBSRequest,
signatureAlgorithm   AlgorithmIdentifier,
signatureValue       OCTET STRING
}
```

其中：

```
KSRequest ::=SEQUENCE {
version              Version DEFAULT v1,
caName               EntName,
taskNO               TaskNO,
reqType              ReqType,
requestList          SEQUENCE OF Request,
requestTime          RequestTime,
requestProof         RequestProof
}
Version ::=INTEGER { v1(0)}
EntName ::=SEQUENCE {
hashAlgorithm        AlgorithmIdentifier,
entName              OCTET STRING,
entPubKeyHash        OCTET STRING,
serialNumber         CertificateSerialNumber
}
TaskNO ::=INTEGER
ReqType ::=CHOICE {
applyKey             INTEGER { apply(11) }
restoreKey           INTEGER { restore(21) }
cancelKey            INTEGER { cancel(31) }
}
Request ::=CHOICE {
applykeyreq          AppKeyReq,
restorekeyreq        RestoreKeyReq,
cancelkeyreq         CancelKeyReq
}
RequestTime ::=GeneralizedTime;
RequestProof ::=OCTET STRING。
```

A.2.2.2 KSRequest 及其结构解释

KSRequest 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.2.2.2.1 **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。

A.2.2.2.2 **请求者标识符**

本项描述了请求者标识符。结构如下：

```
EntName ::=SEQUENCE {
hashAlgorithm          AlgorithmIdentifier,
entName                OCTET STRING,
entPubKeyHash          OCTET STRING,
serialNumber           CertificateSerialNumber
}
```

entName是申请者的唯一名称，该值由运行CA和KMC约定。

entPubKeyHash是申请者公钥的摘要值。该值将通过对发布者证书中的主体公钥字段(不含标记和长度)进行计算。hashAlgorithm字段用来指明这些摘要计算所使用的数据摘要算法。

serialNumber是申请者的证书序列号。

A.2.2.2.3 **任务序列号**

本项描述了请求的任务序列号，该任务序列号是申请者用来区分多次申请时候的一个标识符。

任务序列号是一个整型值，KMC应能处理不大于20字节的任务序列号，而CA应确保不使用大于20字节的任务序列号。

A.2.2.2.4 **请求类型**

本项描述了请求包类型，当值为applyKey时，表明该请求为申请密钥，为restoreKey时表明该请求为恢复密钥，值为cancelKey时表明该请求为撤销密钥。

本项的取值决定了下面一项——详细请求子包Request的取值。

A.2.2.2.5 **详细请求子包**

本项描述申请者请求中的详细请求子包，每个子包的格式如下：

```
Request ::=CHOICE {
applykeyreq          ApplyKeyReq,      当ReqType取值applyKey时
restorekeyreq        RestoreKeyReq,    当ReqType取值restoreKey时
cancelkeyreq         CancelKeyReq      当ReqType取值cancelKey时
}
```

上面三种数据格式解释如下：

a) ApplyKeyReq包

ApplyKeyReq包为密钥申请格式包，当ReqType取值applyKey时，该请求包采用本子包格式，其具体格式如下：

```
ApplyKeyReq ::=SEQUENCE {
appKeyType       AlgType,
appKeyLen        AppKeyLen,
retAsymAlg       AlgType,
retSymAlg        AlgType,
retHashAlg       AlgType,
isRetPubKeyEnv   bool,
appUserInfo      AppUserInfo
}
```

AlgType ::=AlgorithmIdentifier，表明使用的非对称算法、对称算法、摘要算法等算法类型。其

中，appKeyType 为要申请的加密密钥对的类型，retAsymAlg、retSymAlg、retHashAlg 分别为 KMC 响应数据包中非对称算法、对称算法、摘要算法类型；

AppKeyLen ::=INTEGER，表示申请的密钥长度，如十进制 1024 表示申请 1024 位长度的密钥；

isRetPubKeyEnv 项指定 KMC 响应数据包中是否对返回公钥进行加密；

AppUserInfo ::=SEQUENCE {

userName　　OCTET STRING，

userCertNo　　CertificateSerialNumber，

userPubKey　　SubjectPublicKeyInfo，

notBefore　　GeneralizedTime，

notAfter　　GeneralizedTime

}

AppUserInfo 结构表示申请包中对应用户信息，依次表示用户姓名、解密证书序列号、用户签名公钥、密钥有效起始时间、密钥截止时间。

b)　RestoreKeyReq 包

RestoreKeyReq 包为密钥恢复格式包，当 ReqType 取值 restoreKey 时，该请求包采用本子包格式，其具体格式如下：

RestoreKeyReq ::=SEQUENCE {

retAsymAlg　　AlgType，

retSymAlg　　AlgType，

retHashAlg　　AlgType，

isRetPubKeyEnv　　bool，

userCertNo　　CertificateSerialNumber，

userPubKey　　SubjectPublicKeyInfo

}

AlgType ::=AlgorithmIdentifier，表明使用的非对称算法、对称算法、摘要算法等算法类型。其中，retAsymAlg、retSymAlg、retHashAlg 分别为 KMC 响应数据包中非对称算法、对称算法、摘要算法类型；

isRetPubKeyEnv 指定 KMC 响应数据包中是否对返回公钥进行加密；

userCertNo 指定用户证书序列号；

userPubKey 指定用户签名公钥。

c)　CancelKeyReq 包

CancelKeyReq 包为密钥撤销格式包，当 ReqType 取值 cancelKey 时，该请求包采用本子包格式，其具体格式如下：

CancelKeyReq ::=SEQUENCE {

userCertNo　　CertificateSerialNumber

}

userCertNo 指定用户证书序列号。

A.2.2.2.6　请求时间

本项描述请求生成时间。

A.2.2.2.7　请求证据

本项描述请求证据。

A.2.3　响应

A.2.3.1　响应数据格式

KMC 响应的基本格式如下：

```
KMRespond ::=SEQUENCE {
ksRespond                TBSRespond,
signatureAlgorithm       AlgorithmIdentifier,
signatureValue           OCTET STRING
}
```

其中：

```
TBSRespond ::=SEQUENCE {
version              Version DEFAULT v1,
kmcName              entName,
taskNO               TaskNO,
respondType          RespondType,
respondList          SEQUENCE OF Respond,
respondTime          RespondTime,
respondProof         RespondProof
}
```

Version、entName、TaskNo 数据格式在前文已经解释。

```
RespondType ::=CHOICE {
applyRespond         INTEGER { apply(12) }
restoreRespond       INTEGER { restore(23) }
cancelRespond        INTEGER { cancel(33) }
}
Respond ::=CHOICE {
applyKeyRespond          AppKeyRespond,
restoreKeyRespond        RestoreKeyRespond,
cancelKeyRespond         CancelKeyRespond
}
RespondTime ::=GeneralizedTime;
RespondProof ::=OCTET STRING。
```

A.2.3.2 KSRespond 及其结构解释

KSRespond 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.2.3.2.1 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。

A.2.3.2.2 响应者标识符

本项描述了响应者标识符，该结构在前文已经给出，在本响应数据中，其成员分别取值响应者的名称、响应者公钥的摘要值、摘要算法以及响应者的证书序列号。

A.2.3.2.3 任务序列号

本项描述了响应的任务序列号，该任务序列号值取自申请者数据包。

A.2.3.2.4 响应类型

本项描述了响应包类型，当值为 applyRespond 时，表明该包为申请密钥响应包，为 restoreRespond 时表明该包为恢复密钥响应包，值为 cancelRespond 时表明该包为撤销密钥响应包。

本项的取值决定了下面一项——详细响应子包 Respond 的取值。

A.2.3.2.5 详细响应子包

本项描述响应者请求中的详细响应子包，每个子包的格式如下：

```
Respond ::=CHOICE {
applyKeyRespond        retKeyRespond,        当 RespondType 取值 applyRespond 时
restoreKeyRespond      retKeyRespond,        当 RespondType 取值 restoreRespond 时
cancelKeyRespond       CancelKeyRespond      当 RespondType 取值 cancelRespond 时
}
```

上面共有两种数据格式，分别解释如下：

a) retKeyRespond 包

retKeyRespond 包为密钥响应格式包，在处理密钥申请、恢复申请时响应申请者的。当 RespondType 取值 applyRespond、restoreRespond 时，该响应包采用本子包格式，其具体格式如下：

```
retKeyRespond ::=SEQUENCE {
userCertNo          CertificateSerialNumber,
isRetPubKeyEnv      bool,
retPubKey           RetPubKeyENV,
retPriKey           dataEnvelope
}
RetPubKeyENV ::=CHOICE {
userEncPubKey           SubjectPublicKeyInfo,
userEncPubKeyEnv        dataEnvelope
}
dataEnvelope ::= SignedAndEnvelopedData (data)
```

userCertNo 指用户加密证书序列号，该项从 CA 申请包中取值；

isRetPubKeyEnv 表明 KMC 响应数据包中返回公钥数据是否已做作加密，若该项取值 0，表明接下来的公钥数据是公钥明文，而不是加密包格式；

retPubKey 是返回给申请者的用户加密公钥数据，其具体采用的格式根据 isRetPubKeyEnv 而定；

retPriKey 是返回给申请者的用户加密私钥数据，即对私钥作带签名的加密。

b) CancelKeyRespond 包

CancelKeyRespond 包为密钥撤销响应格式包，在处理密钥撤销时响应申请者的。当 RespondType 取值 cancelRespond 时，该响应包采用本子包格式，其具体格式如下：

```
CancelKeyRespond ::=SEQUENCE {
userCertNo        CertificateSerialNumber,
}
```

userCertNo 指定用户加密证书序列号，该项值取自申请包。

A.2.3.2.6 响应时间

本项描述响应生成时间。

A.2.3.2.7 响应证据

本项描述响应证据。

A.2.4 回执(本项为可选项)

A.2.4.1 回执数据格式

接受到 KMC 响应数据后，申请者制作回执数据包，并发送给 KMC，该回执包基本格式如下：

```
CaReceipt ::=SEQUENCE {
ksReceipt              TBSReceipt,
signatureAlgorithm     AlgorithmIdentifier,
signatureValue         OCTET STRING
```

```
}
其中：
KSReceipt ::=SEQUENCE {
version             Version DEFAULT v1,
caName              entName,
taskNO              TaskNO,
receiptType         ReceiptType,
ReceiptUserList     SEQUENCE OF ReceiptUser,
receiptTime         ReceiptTime,
receiptProof        ReceiptProof
}
Version、entName、TaskNo 数据格式在前文已经解释。
ReceiptType ::=CHOICE {
applyRecipt         INTEGER { apply(13) }
restoreRecipt       INTEGER { restore(23) }
cancelRecipt        INTEGER { cancel(33) }
}
ReceiptUser ::=SEQUENCE {
userCertNo          CertificateSerialNumber,
}
RespondTime ::=GeneralizedTime;
RespondProof ::=OCTET STRING。
```

A.2.4.2 KSReceipt 及其结构解释

KSReceipt 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.2.4.2.1 版本

本项描述了回执语法的版本号，当前版本为 1，取整型值 0。

A.2.4.2.2 申请者标识符

本项描述了申请者标识符，该结构和解释在前文已经给出。

A.2.4.2.3 任务序列号

本项描述了回执的任务序列号，该任务序列号值取自响应数据包。

A.2.4.2.4 回执类型

本项描述了回执包类型，当值为 applyRecipt 时，表明该包为申请密钥回执包，为 restoreRecipt 时表明该包为恢复密钥回执包，值为 cancelRecipt 时表明该包为撤销密钥回执包。

A.2.4.2.5 用户信息包

本项描述回执中的用户信息，包括用户加密证书序列号。

A.2.4.2.6 回执时间

本项描述回执生成时间。

A.2.4.2.7 回执证据

本项描述回执证据。

附 录 B
（资料性附录）
安全通信协议

B.1 符号说明

A：在身份鉴别协议中是验证方，在密钥交换协议中是发起方；
B：在身份鉴别协议中是被验证方，在密钥交换协议中是接受方；
$E_K(M)$，$D_K(M)$：使用密钥 K 对消息 M 进行加密和解密；
$S_K(M)$：使用密钥 K 对消息 M 进行签名；
P1_X，S1_X：X 加/解密用的公钥和私钥；
S2_X，P2_X：X 签名/验证用的私钥和公钥；
$ECert_X$，$SCert_X$：X 的加密证书和签名证书；
R_X，N_X：X 产生的随机数和 X 任意选择的信息；
N_X'：X 任意选择的另一信息；
IDn：第 n 个数据包的标识符，n 是自然数；
algId：签名算法；
KEA：密钥交换算法；
K：随机密钥。

B.2 身份鉴别

1. $A \rightarrow B$：{ID_1，R_A，N_A}；
2. $B \rightarrow A$：{ID_2，$SCert_B$，(R_A，R_B，N_B)，$S_{S2\_B}(R_A, R_B, N_B)$}；
3. A 验证 R_A，$SCert_B$ 和 $S_{S2\_B}(R_A, R_B, N_B)$；
   $A \rightarrow B$：{ID_3，$SCert_A$，(R_B，R_A，N_A')，$S_{S2\_A}(R_B, R_A, N_A')$}；
4. B 验证 R_B，$SCert_A$ 和 $S_{S2\_A}(R_B, R_A, N_A')$。

B.3 密钥交换

以下协议陈述中出现的信息包 STP-REQ-TOKEN 和 STP-REP-IT-TOKEN 的格式在 3 中定义。
1. $A \rightarrow B$：{STP-REQ-TOKEN}；
信息包 STP-REQ-TOKEN 中有关各字段的内容赋值如下：
randSrc：$=R_A$，targ-name：=B，src-name：=A，key-estb-set：=*KEA*，
algId：=*algId*，req-integrity：$=S_{S2\_A}(R_A, B, A, KEA)$，certif-data：$=ECert_A$。
2. B 验证 $ECert_A$ 和 $S_{S2\_A}(R_A, B, A, KEA)$；$B \rightarrow A$：{STP-REP-IT-TOKEN}；
信息包 STP-REP-IT-TOKEN 中有关各字段的内容赋值如下：
randSrc：$=R_A$，randTarg：$=R_B$，targ-name：=B，src-name：=A，
key-estb-id：=*KEA*，key-estb-str：$=E_{P1\_A}(K)$，algId：=*algId*，
rep-ti-integ：$=S_{S2\_B}(R_A, R_B, B, A, KEA, E_{P1\_A}(K))$，certif-data：$=ECert_B$；
3. A 验证 $ECert_B$，$S_{S2\_B}(R_A, R_B, B, A, KEA, E_{P1\_A}(K))$ 和 R_A，计算 $D_{S1\_A}(E_{P1\_A}(K))=K$，
$A \rightarrow B$：{STP-REP-IT-TOKEN}；
信息包 STP-REP-IT-TOKEN 中有关各字段的内容赋值如下：
randSrc：$=R_A$，

randTarg：=R_B，

targ-name：=B，

src-name：=A，

algId：=algId

key-estb-rep：=$E_{P1\_B}(K,R_B)$

rep-it-integ：=$S_{S2\_A}(R_A,R_B,B,A,E_{P1\_B}(K,R_B))$；

4. B 验证 $ECert_A$ 和 $S_{S2\_A}(R_A,R_B,B,A,E_{P1\_B}(K,R_B))$，计算 $D_{S1\_B}(E_{P1\_B}(K,R_B))=(K,R_B)$，验证 K 和 R_B。

B.4 安全通信协议

验证方请求：

```
STP-REQ ::=SEQUENCE {
        RequestToken        REQ-TOKEN 请求数据包
        certif-data [0]     CertificationData OPTIONAL 证书
        auth-data [1]       AuthorizationData OPTIONAL
}
REQ-TOKEN ::=SEQUENCE {
        req-contents   Req-contents  请求的内容
        algId          AlgorithmIdentifier 完整性算法
        req-integrity  Integrity  --"token" is Req-contents 请求的内容的完整性验证
}
Req-contents ::=SEQUENCE {
        tok-id          INTEGER (256)          TOKEN 的标识
        context-id      Random-Integer         上下文标识
        pvno            BIT STRING             协议版本
        timestamp       UTCTime OPTIONAL       时间戳
        randSrc         Random-Integer         随机数(A)
        targ-name       Name                   对方的名字
        src-name [0]    Name OPTIONAL          本方的名字，除"匿名"外，必须支持
        req-data        Context-Data           上下文数据
        validity [1]    Validity OPTIONAL      上下文的有效期
        key-estb-set    Key-Estb-Algs          密钥交换算法集合
        key-estb-req    BIT STRING OPTIONAL第一个密钥交换算法的参数，这个参数必
须包括密钥的长度。如果发起方不想进行密钥交换除外
        key-src-bind    OCTET STRING OPTIONAL 密钥和本方名字的绑定，强制的 Hash
函数过程
        }
```

被验证方回应：

```
STP-REP-TI ::=SEQUENCE {
        ResponseToken   REP-TI-TOKEN  回应数据包
        certif-data     CertificationData OPTIONAL  证书
}
```

REP-TI-TOKEN ::=SEQUENCE {

rep-ti-contents	Rep-ti-contents 回应数据包的内容
algId	AlgorithmIdentifier 完整性算法
rep-ti-integ	Integrity--"token" is Rep-ti-contents 回应内容的完整性验证

}

Rep-ti-contents ::=SEQUENCE {

tok-id	INTEGER (512) TOKEN 的标识
context-id	Random-Integer 上下文标识
pvno [0]	BIT STRING OPTIONAL 协议版本
timestamp	UTCTime OPTIONAL 时间戳
randTarg	Random-Integer 随机数(B)
src-name [1]	Name OPTIONAL 验证方的名字
targ-name	Name 被验证方的名字
randSrc	Random-Integer 随机数(A)
rep-data	Context-Data 上下文数据
validity [2]	Validity 上下文的有效期,是 REQ-TOKEN 中上下文的有效期的子集
key-estb-id	AlgorithmIdentifier OPTIONAL 密钥交换算法,是 REQ-TOKEN 中密钥交换算法集合中的一个
key-estb-str	BIT STRING OPTIONAL 密钥建立信息

}

验证方回应:

STP-REP-IT ::=SEQUENCE {

responseToken	REP-IT-TOKEN 回应数据包
algId	AlgorithmIdentifier 完整性算法
rep-it-integ	Integrity--"token" is REP-IT-TOKEN 回应数据包的完整性验证

}

REP-IT-TOKEN ::=SEQUENCE {

tok-id	INTEGER (768) TOKEN 的标识
context-id	Random-Integer 上下文标识
randSrc	Random-Integer 随机数(A)
randTarg	Random-Integer 随机数(B)
targ-name	Name 被验证方的名字
src-name	Name OPTIONAL 验证方的名字
key-estb-rep	BIT STRING OPTIONAL 回应密钥建立信息

}

附 录 C
（资料性附录）
密码设备接口函数定义及说明

C.1 应用类密码设备接口函数

C.1.1 接口组成部分

C.1.1.1 算法对象

在接口函数中，使用 ALGORITHM_OBJ 表示算法对象，用来保持算法的参数的信息和在密码计算中保持上下文的关系。在调用密码功能之前，必须调用 M_CreateAlgorithmObject 创建算法对象，调用 M_SetAlgorithmObject 设置算法对象。每一个算法对象必须调用 M_CreateAlgorithmObject 函数创建，调用 M_DestroyAlgorithmObject 函数销毁。一个算法对象不能既用于加密也用于解密，只能或者加密或者解密。一旦算法对象被设置，将不能被重新设置。对于一个算法对象，不能调用 M_SetAlgorithmInfo 函数两次。或者创建一个新的对象，或者销毁对象然后重新创建。

C.1.1.2 密钥对象

在接口函数中使用 KEY_OBJ 表示密钥对象，这个参数用来保持密钥的值。在调用密码功能之前，必须调用 M_CreateKeyObject 创建密钥对象，调用 M_SetKeyObject 设置密钥对象。每一个密钥对象必须调用 M_CreateKeyObject 函数创建，调用 M_DestroyKeyObject 函数销毁。

C.1.1.3 算法函数

使用算法对象和密钥对象调用相应的密码算法函数执行密码的功能。

C.1.2 宏定义

```
#ifndef NULL
#define NULL    0
#endif
#ifndef NULL_PTR
#define NULL_PTR    NULL
#endif          /*定义空*/
typedef unsigned char *POINTER;          /*定义指针*/
typedef unsigned long INFO_TYPE;     /*定义算法信息的类型*/
typedef POINTER    ALGORITHM_OBJ;          /*定义算法对象*/
typedef POINTER    KEY_OBJ;          /*定义密钥对象*/
#define CKI_SymmetricKey    1          /*对称密钥*/
#define CKI_CipherSymmetricKey    2          /*加密的对称密钥*/
#define CKI_RSAPrivateRef    10          /*RSA 私钥的标识*/
#define CKI_RSAPrivateCipher    11          /*加密的 RSA 的私钥*/
#define CKI_RSAPrivateDER    12          /*RSA 私钥的 DER 编码*/
#define CKI_RSAPublicDER    15          /*RSA 公钥的 DER 编码*/
#define CKI_X9_ECPrivateDER    20          /*EC 私钥的 DER 编码*/
#define CKI_X9_ECPublicDER    25          /*EC 公钥的 DER 编码*/
#define CAI_SCH    101          /*SCH 算法*/
#define CAI_HMAC    103          /*使用带有密钥的 HASH 函数计算消息认证码*/
#define CAI_ECKeyGen    104          /*产生 ECC 密钥对*/
```

```
#define CAI_RSAKeyGen    105      /*产生 RSA 的密钥对,公钥和私钥都以 DER 编码的形式输出*/
#define CAI_RSATokenKeyGen    106      /*产生 RSA 的密钥对,公钥以 DER 编码的形式输出,私钥以引用的形式输出*/
#define CAI_RSASecretKeyGen    107      /*产生 RSA 的密钥对,公钥以 DER 编码的形式输出,私钥以加密的形式输出*/
#define CAI_SCHWithRSAEncryption    109      /*使用 RSA 和数据摘要数字签名*/
#define CAI_EC_SCHWithDSA    111      /*使用 ECDSA 数字签名*/
#define CAI_SymmetricKeyGen    112      /*产生对称密钥的功能*/
#define CAI_SymmetricCipher    113      /*使用对称算法加密和解密*/
#define CAI_RSAEncryption    114      /使用 RSA 算法加密和解密*/
#define CAI_Random    115    /*产生随机数*/
#define CAI_EC_DHKeyAgree    116      /*基于 ECDH 算法的密钥交换算法*/
```

注：本部分中引用的"SCH",是指国家密码主管部门批准使用的数据摘要算法。

C.1.3 数据结构定义

C.1.3.1 数据元

```
typedef struct  element_struct{
    unsigned char *data;
    unsigned long  len;
} ELEMENT;
/*定义通用的数据单元*/
```

C.1.3.2 数据摘要算法参数

```
typedef struct {
    INFO_TYPE digestInfoType;  /*文摘算法的类型*/
    POINTER digestInfoParams;  /*文摘算法的参数*/
  } A_DIGEST_SPECIFIER;
```

C.1.3.3 RSA 密钥参数

```
typedef struct {
    unsigned long modulusBits;    /*模的长度 */
    ELEMENT publicExponent;    /*公钥的指数 */
  } A_RSA_KEY_GEN_PARAMS;
typedef struct {
    unsigned long keyReference;    /*密钥的引用 */
  } A_TOKEN_KEY_GEN_PARAMS;
```

C.1.3.4 椭圆曲线参数的类型

```
Enum    A_EC_ParameterType{
          EC_PARAMS_TYPE_NULL=100,      /*类型是空*/
          EC_PARAMS_TYPE_CURVE,         /*类型是曲线*/
          EC_PARAMS_TYPE_NAMED_CURVE /*类型是命名曲线*/
}A_EC_PARAMS_TYPE;
/*定义椭圆曲线参数的类型*/
```

C.1.3.5 椭圆曲线域的类型

```
Enum    A_EC_FieldType{
```

```
        EC_ FT_FP=100,                    /* 素数域 */
        EC_ FT_F2_POLYNOMIAL,             /* 二进制域多项式方式 */
        EC_ FT_F2_ONB                     /* 二进制域优化方式 */
    }A_EC_FIELD_TYPE;
    /* 定义椭圆曲线的域的类型 */
```

C.1.3.6 椭圆曲线参数

```
    typedef struct {
        A_EC_PARAMS_TYPE parameterInfoType;/* 曲线参数的类型 */
        POINTER parameterInfoValue;/* 曲线参数的值 */
      } A_EC_PARAMS;
    /* 定义曲线的参数 */
```

如果曲线参数的类型是 EC_PARAMS_TYPE_NULL，曲线参数的值设置为 NULL_PTR。

如果曲线参数的类型是 EC_PARAMS_TYPE_CURVE，曲线参数的值设置为指向结构 A_EC_PARAMS_CURVE 的指针。

如果曲线参数的类型是 EC_PARAMS_TYPE_NAMED_CURVE，曲线参数的值设置为指向结构 ELEMENT 的指针。结构 ELEMENT 的 data 字段是命名曲线的 Object identifier 的 DER 编码，结构 ELEMENT 的 len 字段是命名曲线的 Object identifier 的 DER 编码的长度。

```
    A_EC _PARAMS_CURVE
        typedef struct {
        unsigned long version;                /* 版本 */
        A_EC_FIELD_TYPE     fieldType;        /* 域类型 */
        ELEMENT fieldInfo;                    /* 域的信息，如果是素数域，信息表示一个素数。
如果是二进制域多项式形式，信息表示一个多项式。 */
        ELEMENT coeffA;                       /* coefficient A */
        ELEMENT coeffB;                       /* coefficient B */
        ELEMENT base;                         /* 基底 */
        ELEMENT order;                        /* 序 */
        ELEMENT cofactor;
        unsigned long fieldElementBits;
      } A_EC_PARAMS_CURVE;
    /* 定义椭圆曲线的参数 */
```

C.1.3.7 对称密钥标识

```
    typedef struct {
        unsigned long keyLength;      /* 密钥的长度，字节为单位 */
        unsigned long protectFlag;    /* 密钥的保护标记，0 表示不加密密钥，非 0 表示加密密钥 */
        unsigned char *cipherName;    /* 算法的名字 */
    } A_SYMMETRIC_KEY_SPECIFIER;
```

C.1.3.8 对称密钥加密算法参数

```
    typedef struct {
        unsigned char *encryptionMethodName; /* 算法的名字 */
        POINTER encryptionParams;            /* 算法的参数 */
        unsigned char *feedbackMethodName;   /* 加密方式的名字 */
```

```
    POINTER feedbackParams;      /* 加密方式的参数,指向 ELEMENT 结构的初始化向量 */
    unsigned char paddingMethodName;      /* 补位方法的名字 */
}A_SYMMETRIC_CIPHER_ PARAMS;
```

C.1.3.9 随机数

```
typedef struct {
    POINTER Seed;          /* 随机数的种子 */
}A_RANDOM_ PARAMS;
```

C.1.4 错误码

```
#define ME_ALGORITHM_ALREADY_SET     512
/* 算法对象已经调用 B_SetAlgorithmInfo 被设置或调用 algorithm parameter generation */

#define ME_ALGORITHM_INFO     513
/* 无效的 Algorithm information 的格式在 algorithm object 算法对象中 */

#define ME_ALGORITHM_NOT_INITIALIZED     514
/* algorithm object 没有被调用初始化过程初始化 */

#define ME_ALGORITHM_NOT_SET     515
/* algorithm object 没有被调用 B_SetAlgprithmInfo 函数设置 */

#define ME_ALGORITHM_OBJ     516
/* 无效的 algorithm object */

#define ME_ALG_OPERATION_UNKNOWN     517
/* 对应的一个算法或算法信息的类型的未知的操作 */

#define ME_ALLOC     518
/* 内存不足 */

#define ME_DATA     520
/* 通常的数据错误 */

#define ME_EXPONENT_EVEN     521
/* 密钥对产生中公钥的指数是无效的偶数的值 */

#define ME_EXPONENT_LEN     522
/* 密钥对产生中公钥的指数是无效的指数的长度 */

#define ME_INPUT_DATA     524
/* 输入数据无效的编码格式 */

#define ME_INPUT_LEN     525
/* 输入的数据的全部的长度无效 */
```

```
#define ME_KEY_ALREADY_SET      526
/* key object 的值已经被 B_SetKeyInfo 函数调用或被 key generation 调用 */

#define ME_KEY_INFO      527
/* 无效的 key information 的格式在 key object */

#define ME_KEY_LEN      528
/* 无效的密钥的长度 */

#define ME_KEY_NOT_SET      529
/* 这个 key object 没有被用 B_SetKeyInfo 函数设置或调用 key generation */

#define ME_KEY_OBJ      530
/* 无效的 key object */

#define ME_KEY_OPERATION_UNKNOWN      531
/* 对应的一个 key info 的类型的未知的操作 */

#define ME_MEMORY_OBJ      532
/* 无效的内部的内存对象 */

#define ME_MODULUS_LEN      533
/* 不支持的模数的长度对于一个密钥或 algorithm parameters */

#define ME_NOT_INITIALIZED      534
/* 算法被不正确的初始化 */

#define ME_NOT_SUPPORTED      535
/* 对应一个指定的算法,算法选择器不支持 key object 中的 key information 的类型 */

#define ME_OUTPUT_LEN      536
/* 用于接受输出的最大的尺寸或输出的 buffer 太小 */

#define ME_RANDOM_NOT_INITIALIZED      538
/* 随机数算法没有被调用 B_RandomInit 函数初始化 */

#define ME_RANDOM_OBJ      539
/* 对应随机数算法的无效的算法对象 */

#define ME_SIGNATURE      540
/* 签名不能被验证 */
```

```
#define ME_WRONG_ALGORITHM_INFO          541
/* 要求的 algorithm information 不在 algorithm object 中 */

#define ME_WRONG_KEY_INFO     542
/* 要求的 key information 不在 key object 中 */

#define ME_INPUT_COUNT     543
/* 对应的输入的数据,Update 被调用一个无效的记数的次数 */

#define ME_OUTPUT_COUNT          544
/* 对应的输出的数据,Update 被调用一个无效的记数的次数 */

#define ME_METHOD_NOT_IN_CHOOSER     545
/* 算法选择器不包括 algorithm method,这个 algorithm 先前被 B_SetAlgorithminfo 函数设置 */

#define ME_KEY_WEAK          546
/* 密钥的数据提供产生一个知道的弱的密钥 */

#define BE_BAD_POINTER     548
/* 无效的指针 */
```

C.1.5 函数定义及说明

C.1.5.1 对称密钥产生函数

M_SymmetricKeyGenerateInit

声明：

```
int M_SymmetricKeyGenerateInit (
  ALGORITHM_OBJ algorithmObject      /* 产生对称密钥的算法对象 */
);
```

解释：

调用 M_SymmetricKeyGenerateInit 函数初始化产生对称密钥的算法对象。这个对象在调用之前必须被 M_SetAlgorithmObject 函数设置。

返回值：

0	成功
非 0	见错误码的定义和说明

M_SymmetricKeyGenerate

声明：

```
int M_SymmetricKeyGenerate (
  ALGORITHM_OBJ algorithmObject,     /* 产生对称密钥的算法对象 */
  KEY_OBJ symmetricKey               /* 新的对称密钥对象 */
);
```

解释：

调用 M_SymmetricKeyGenerate 函数产生一个对称密钥,并且输出结果保存在参数 symmetricKey

中。参数 symmetricKey 指明的密钥对象，在调用此函数之前必须调用 M_CreateKeyObject 函数创建。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.2　产生公钥密钥对函数

M_PublicKeyGenerateInit

声明：

```
int M_PublicKeyGenerateInit (
  ALGORITHM_OBJ algorithmObject     /* 产生密钥对的算法对象 */
);
```

解释：

调用 M_PublicKeyGenerateInit 函数初始化产生非对称密钥对的算法对象。这个对象在调用之前必须被 M_SetAlgorithmObject 函数设置。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_GeneratePublicKeyPair

声明：

```
int M_GeneratePublicKeypair (
  ALGORITHM_OBJ algorithmObject,    /* 产生密钥对的算法对象 */
  KEY_OBJ publicKey,                /* 新的公钥对象 */
  KEY_OBJ privateKey                /* 新的私钥对象 */
  );
```

解释：

调用 M_GeneratePublicKeyPair 函数产生非对称密钥对，并且输出公钥和私钥中。参数 publicKey 指明的公钥对象，在调用此函数之前必须调用 M_CreatePublicKeyObject 函数创建。参数 privateKey 指明的私钥对象，在调用此函数之前必须调用 M_CreatePublicKeyObject 函数创建。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.3　加密函数

M_EncryptInit

声明：

```
int M_EncryptInit (
  ALGORITHM_OBJ algorithmObject,    /* 算法对象 */
  KEY_OBJ keyObject                 /* 密钥对象 */
);
```

解释：

M_EncryptInit 函数初始化加密数据使用的算法对象。这个对象先前必须被 M_SetAlgorithmObject 函数调用。参数 keyObject 指明的密钥对象提供密钥的信息。

M_EncryptInit 调用一次设置算法和密钥，M_EncryptUpdate 调用多次加密数据，M_EncryptFinal 调用一次处理最后的分组包括补位的字节。

在调用 M_EncryptFinal 以后，可以调用 M_EncryptUpdate 函数处理另外的数据，

如果采用 CBC 的方式，并且使用不同的初始化向量(IV)，在调用 M_EncryptUpdate 函数之前必须调用 M_SetAlgorithmObject 函数设置新的 IV。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_EncryptUpdate

声明：

```
int M_EncryptUpdate (
    ALGORITHM_OBJ algorithmObject,          /* 算法对象 */
    unsigned char *pDataOut,                /* 输出数据 */
    unsigned long *pDataOutLen,             /* 输出数据的长度 */
    unsigned long maxPartOutLen,            /* 输出数据缓冲区的大小 */
    unsigned char *pDataIn,                 /* 输入数据 */
    unsigned long partInLen                 /* 输入数据的长度 */
    );
```

解释：

M_EncryptUpdate 函数用来加密数据并输出结果。输出的结果的内容最多输出参数 maxPartOutLen 指定的长度。输出结果保存在参数 pDataOut 中，并且输出结果的长度在参数 pDataOutLen 中。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_EncryptFinal

声明：

```
int M_EncryptFinal (
    ALGORITHM_OBJ algorithmObject,      /* 算法对象 */
    unsigned char *pDataOut,            /* 输出数据 */
    unsigned long *pDataOutLen,         /* 输出数据的长度 */
    unsigned long maxDataOutLen         /* 输出数据缓冲区的大小 */
);
```

解释：

M_EncryptFinal 函数完成加密数据最后的过程并输出结果。输出的结果的内容最多输出参数 maxDataOutLen 指定的长度。输出结果保存在参数 pDataOut 中，并且输出结果的长度在参数 pDataOutLen 中。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.4　解密函数

M_DecryptInit

声明：

```
int M_DecryptInit (
```

```
    ALGORITHM_OBJ algorithmObject,      /*算法对象 */
    KEY_OBJ keyObject                   /*密钥对象 */
  );
```

解释：

M_DecryptInit 函数初始化解密数据使用的算法对象。这个对象先前必须被 M_SetAlgorithmObject 函数调用。参数 keyObject 指明的密钥对象提供密钥的信息。

M_DecryptInit 调用一次设置算法和密钥，M_DecryptUpdate 调用多次解密数据，M_DecryptFinal 调用一次处理最后的分组包括补位的字节。

在调用 M_DecryptFinal 以后，可以调用 M_DecryptUpdate 函数处理另外的数据，

如果采用 CBC 的方式，并且使用不同的初始化向量(IV)，在调用 M_DecryptUpdate 函数之前必须调用 M_SetAlgorithmObject 函数设置新的 IV。

返回值：

0	成功
非 0	见错误码的定义和说明

M_DecryptUpdate

声明：

```
int M_DecryptUpdate (
    ALGORITHM_OBJ algorithmObject,      /*算法对象 */
    unsigned char *pDataOut,            /*输出数据 */
    unsigned long *pDataOutLen,         /*输出数据的长度 */
    unsigned long maxDataOutLen,        /*输出数据缓冲区的大小 */
    unsigned char *pDataIn,             /*输入数据 */
    unsigned long DataInLen             /*输入数据的长度 */
    );
```

解释：

M_DecryptUpdate 函数用来解密数据并输出结果。输出的结果的内容最多输出参数 maxDataOutLen 指定的长度。输出结果保存在参数 pDataOut 中，并且输出结果的长度在参数 pDataOutLen 中。

返回值：

0	成功
非 0	见错误码的定义和说明

M_DecryptFinal

声明：

```
int M_DecryptFinal (
    ALGORITHM_OBJ algorithmObject,  /*算法对象*/
    unsigned char *pDataOut,        /*输出数据*/
    unsigned long *DataOutLen,      /*输出数据的长度*/
    unsigned long maxDataOutLen     /*输出数据缓冲区的大小*/
);
```

解释：

M_DecryptFinal 函数完成解密数据最后的过程并输出结果。输出的结果的内容最多输出参数

maxDataOutLen 指定的长度。输出结果保存在参数 pDataOut 中，并且输出结果的长度在参数 pDataOutLen 中。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.5　签名函数

M_SignInit

声明：

```
int M_SignInit (
    ALGORITHM_OBJ algorithmObject,      /*算法对象 */
    KEY_OBJ keyObject,                  /*密钥对象 */
);
```

解释：

M_SignInit 函数初始化数字签名使用的算法对象。这个对象先前必须被 M_SetAlgorithmObject 函数调用。参数 keyObject 指明的密钥对象提供密钥的信息。

M_SignInit 调用一次设置算法和密钥，M_SignUpdate 调用多次处理数据，M_SignFinal 调用一次处理最后的分组包括调用 M_SignUpdate 生产的结果。并输出数字签名的结果。

在调用 M_SignFinal 以后，可以调用 M_SignUpdate 函数处理另外的数据，而不需要再一次调用 M_SignInit 函数。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_SignUpdate

声明：

```
int M_SignUpdate (
    ALGORITHM_OBJ algorithmObject,      /*算法对象*/
    unsigned char *pDataIn,             /*输入的数据 */
    unsigned long DataInLen             /*输入数据的长度 */
);
```

解释：

M_SignUpdate 函数处理待签名的原文数据。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_SignFinal

声明：

```
int M_SignFinal (
    ALGORITHM_OBJ algorithmObject,  /*算法对象 */
    unsigned char *pSignature,      /*输出的数字签名*/
    unsigned long *pSignatureLen,   /*输出的数字签名的长度*/
    unsigned long maxSignatureLen   /*输出的数字签名的缓冲区的大小*/
    );
```

解释：

M_SignFinal 函数完成数字签名最后的过程并输出结果。输出的结果的内容最多输出参数 maxSignatureLen 指定的长度。输出结果保存在参数 signature 中，并且输出结果的长度在参数 signatureLen 中。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.6　验证签名函数

M_VerifyInit

声明：

```
int M_VerifyInit (
    ALGORITHM_OBJ algorithmObject,        /*算法对象 */
    KEY_OBJ keyObject                     /*密钥对象 */
    );
```

解释：

M_VerifyInit 函数初始化验证数字签名使用的算法对象。这个对象先前必须被 M_SetAlgorithmObject 函数调用。参数 keyObject 指明的密钥对象提供密钥的信息。

M_VerifyInit 调用一次设置算法和密钥，M_VerifyUpdate 调用多次处理数据，M_VerifyFinal 调用一次验证数字签名。要验证的数字签名通过 M_VerifyFinal 函数的参数传递。

在调用 M_VerifyFinal 以后，可以调用 M_VerifyUpdate 函数处理另外的数据，而不需要再一次调用 M_VerifyInit 函数。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_VerifyUpdate

声明：

```
int M_VerifyUpdate (
    ALGORITHM_OBJ algorithmObject,   /*算法对象 */
    unsigned char *pDataIn,          /*输入的数据 */
    unsigned long DataInLen          /*输入的数据的长度 */
    );
```

解释：

M_VerifyUpdate 函数处理验证签名的原文数据。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_VerifyFinal

声明：

```
int M_VerifyFinal (
    ALGORITHM_OBJ algorithmObject,        /*算法对象 */
    unsigned char *pSignature,            /*验证的签名 */
```

```
    unsigned long signatureLen                    /* 验证的签名的长度 */
    );
```

解释：

M_VerifyFinal 完成验证数字签名。要验证的数字签名通过 M_VerifyFinal 函数的参数传递。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.7 文摘函数

M_DigestInit

声明：

```
int M_DigestInit (
   ALGORITHM_OBJ algorithmObject,          /* 算法对象 */
   KEY_OBJ keyObject                       /* 密钥对象 */
);
```

解释：

M_DigestInit 函数用来初始化计算消息文摘的算法对象和计算消息文摘的算法，这个算法对象必须先前被 B_SetAlgorithmInfo 函数调用。参数 keyObject 支持密钥的信息，如果参数 keyObject 设置为 NULL_PTR，支持无密钥的文摘算法。如果参数 keyObject 设置为非空，支持有密钥的文摘算法，这个密钥对象必须先前被 B_SetKeyInfo 函数调用。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_DigestUpdate

声明：

```
int   M_DigestUpdate(
   ALGORITHM_OBJ algorithmObject,   /* 算法对象 */
   unsigned char *pDataIn,          /* 输入的数据 */
   unsigned long pDataIn            /* 输入数据的长度 */
);
```

解释：

M_DigestUpdate 函数使用输入的明文更新 algorithmObject 对象。明文的内容是参数 pDataIn，明文的长度是参数 DataInLen。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_DigestFinal

声明：

```
int     M_DigestFinal(
    ALGORITHM_OBJ algorithmObject,/* 算法对象 */
    unsigned char *pDigest,       /* 输出文摘的缓冲区 */
    unsigned long *pDigestLen,    /* 文摘的长度 */
```

```
    unsigned long maxDigestLen          /* 输出文摘的缓冲区的长度 */
);
```

解释：

M_DigestFinal 函数用来最后完成参数 algorithmObject 的文摘处理过程，并且输出文摘。输出的文摘的内容最多输出调用者参数 maxDigestLen 指定的长度。并且输出文摘的长度在参数 pDigestLen 中。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.8 密钥交换函数

M_KeyAgreeInit

声明：

```
int  M_KeyAgreeInit (
    ALGORITHM_OBJ algorithmObject,  /* 算法对象 */
    KEY_OBJ keyObject               /* 密钥对象 */
    );
```

解释：

M_KeyAgreeInit 函数用来初始化密钥交换的算法对象和密钥信息。这个算法对象必须先前被 B_SetAlgorithmInfo 函数调用。参数 keyObject 提供自己的私钥的信息。先调用 M_CreateKeyObject 函数创建一个新的密钥对象。然后调用 M_SetKeyInfo 函数设置密钥的值。

此函数支持 CAI_EC_DHKeyAgree 算法。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_KeyAgree

声明：

```
int  M_KeyAgree (
    ALGORITHM_OBJ algorithmObject,  /* 算法对象 */
    KEY_OBJ keyObject,              /* 密钥对象 */
    unsigned char *pOutput,         /* 输出的数据 */
    unsigned long *outputLen,       /* 输出的数据的长度 */
    unsigned long maxOutputLen,     /* 输出的数据的缓冲区的大小 */
    );
```

解释：

M_KeyAgree 函数完成密钥交换的过程。并且输出结果。输出的结果的内容最多输出调用者参数 maxOutputLen 指定的长度。并且输出结果的长度在参数 outputLen 中。参数 keyObject 提供对方的公钥的信息。先调用 M_CreateKeyObject 函数创建一个新的密钥对象。然后调用 M_SetKeyInfo 函数设置密钥的值。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.9 随机数或伪随机数函数

M_RandomInit

声明：

```
int  M_RandomInit (
  ALGORITHM_OBJ randomAlgorithm   /* 随机数算法对象 */
);
```

解释：

调用 M_RandomInit 初始化随机数算法对象产生随机数。使用的随机数算法对象是先前调用 M_SetAlgorithmInfo 函数设置的随机数算法对象。在随机数算法对象中，如果没有种子的值，M_RandomInit 函数设置一个缺省的种子用来产生随机数。

返回值：

0	成功
非 0	见错误码的定义和说明

M_GenerateRandomBytes

声明：

```
int  M_GenerateRandomBytes (
  ALGORITHM_OBJ randomAlgorithm,   /* 随机数算法对象 */
  unsigned char *pOutput,          /* 输出的缓冲区 */
  unsigned long outputLen          /* 输出的缓冲区的长度 */
);
```

解释：

调用 M_GenerateRandomBytes 产生参数 outputLen 指定长度的伪随机数，并且输出结果。参数 randomAlgorithm 的算法对象必须有种子。

返回值：

0	成功
非 0	见错误码的定义和说明

C.1.5.10 算法对象函数

M_CreateAlgorithmObject

声明：

```
int  M_CreateAlgorithmObject (
  ALGORITHM_OBJ *pAlgorithmObject   /* 新的算法对象 */
);
```

解释：

调用 M_CreateAlgorithmObject 分配和初始化一个新的算法对象。保存结果在参数 pAlgorithmObject 中。如果 M_CreateAlgorithmObject 不成功，没有内存分配给 pAlgorithmObject，设置 pAlgorithmObject 为 NULL_PTR。

返回值：

0	成功
非 0	见错误码的定义和说明

M_DestroyAlgorithmObject

声明：

```
void M_DestroyAlgorithmObject (
```

```
    ALGORITHM_OBJ *pAlgorithmObject   /*算法对象的指针 */
  );
```

解释：

调用 M_DestroyAlgorithmObject 销毁算法对象，算法对象的信息归零。释放算法对象占有的内存。设置 pAlgorithmObject 参数为 NULL_PTR.。如果参数 pAlgorithmObject 已经是 NULL_PTR 或不是一个有效的密钥对象，将不处理这个对象。

这个过程调用以后，所有和这个对象相关的信息都将被阻塞。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_GetAlgorithmInfo

声明：

```
int  M_GetAlgorithmInfo (
    POINTER *info,                    /*算法信息 */
    ALGORITHM_OBJ algorithmObject,    /*算法对象 */
    INFO_TYPE infoType                /*算法的类型 */
  );
```

解释：

调用 M_GetAlgorithmInfo 获取和算法类型相匹配的指定的算法对象的算法参数的信息。算法参数的格式是指定的算法的类型确定的。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_SetAlgorithmInfo

声明：

```
int  M_SetAlgorithmInfo (
    ALGORITHM_OBJ algorithmObject,   /*算法对象 */
    INFO_TYPE infoType,              /*算法对象的类型 */
    POINTER info                     /*算法信息 */
  );
```

解释：

调用 M_SetAlgorithmInfo 设置和算法类型相匹配的指定算法对象的算法参数。算法参数的格式是指定的算法的类型确定的。M_SetAlgorithmInfo 函数将设置的算法的参数的信息拷贝一个分离的副本分配给指定的算法对象。一旦算法对象被设置，将不能被重新设置。对于一个算法对象，不能调用 M_SetAlgorithmInfo 函数两次。或者创建一个新的对象，或者销毁对象然后重新创建。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.1.5.11　密钥对象函数

M_CreateKeyObject

声明：

```
int  M_CreateKeyObject (
```

```
    KEY_OBJ * pKeyObject        /* 新的密钥对象 */
);
```

解释：

调用 M_CreateKeyObject 分配和初始化一个新的密钥对象。保存结果在参数 pKeyObject 中。如果 M_CreateKeyObject 不成功，没有内存分配给 pKeyObject，设置 pKeyObject 为 NULL_PTR。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_DestroyKeyObject

声明：

```
void M_DestroyKeyObject (
    KEY_OBJ * pKeyObject        /* key object 的指针 */
);
```

解释：

调用 M_DestroyKeyObject 销毁密钥对象，密钥对象的信息归零。释放密钥对象占有的内存。设置 pKeyObject 参数为 NULL_PTR. 。如果参数 pKeyObject 已经是 NULL_PTR 或不是一个有效的密钥对象，将不处理这个对象。

这个过程调用以后，所有和这个对象相关的信息都将被阻塞。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_GetKeyInfo

声明：

```
int  M_GetKeyInfo (
    POINTER * pInfo,            /* 密钥的信息 */
    KEY_OBJ keyObject,          /* 密钥对象 */
    INFO_TYPE infoType          /* 密钥信息的类型 */
);
```

解释：

调用 M_GetKeyInfo 获取和密钥信息类型相匹配的指定密钥对象的密钥信息。

返回值：

0　　成功

非 0　　见错误码的定义和说明

M_SetKeyInfo

声明：

```
int  M_SetKeyInfo (
    KEY_OBJ keyObject,          /* 密钥对象 */
    INFO_TYPE infoType,         /* 密钥的类型 */
    POINTER info                /* 密钥的信息 */
);
```

解释：

调用 M_SetKeyInfo 设置和密钥信息类型相匹配的指定密钥对象的密钥信息。M_SetKeyInfo 函数将设置的密钥的信息拷贝一个分离的副本分配给指定的密钥对象。一旦密钥对象被设置，将不能被重新设置。对于一个密钥对象，不能调用 M_SetKeyInfo 函数两次。或者创建一个新的对象，或者销毁对象然后重新创建。

返回值：

0　　成功

非 0　　见错误码的定义和说明

C.2　证书载体接口函数

C.2.1　宏定义

```
#ifndef NULL
#define NULL    0
#endif
#ifndef NULL_PTR
#define NULL_PTR    NULL
#endif
typedef C_HANDLE    HANDLE;        /*定义设备打开的句柄*/
typedef unsigned char *C_KEY_HANDLE;        /*定义密钥的句柄*/
#define CDA_KEY_USAGE_SIGN    1
#define CDA_KEY_USAGE_ENCRYPT    2
#define CDA_KEY_USAGE_KEYAGREE  4
typedef unsigned long CDA_KEYUSAGE;        /*定义密钥的用法*/
```

C.2.2　数据结构定义

```
typedef struct{
        unsigned short year;
        unsigned short month;
        unsigned short day;
        unsigned short hour;
        unsigned short minute;
        unsigned short second;
    }CDA_TIME;
typedef struct{
        CDA_TIME start;          /*开始的时间*/
        Unsigned long elapses;     /*时间的流逝以秒为单位*/
}CDA_PeriodOfValid;
#define     CDA_CERT_TYPE_SIGN    1
#define     CDA_CERT_TYPE_ENCRYPT    2
typedef     unsigned long CDA_Certificate;          /*定义证书的类型*/
typedef struct{
        CDA_Certificate cert;
        Long maxCertLength;/*支持的一个证书文件的最大的空间*/
}CDA_CerticateCapability;                    /*定义支持证书的属性，如果 cert 字段设置为
CDA_CERT_TYPE_SIGN|CDA_CERT_TYPE_ENCRYPT 表示支持双证书 */
```

```
typedef struct{
        unsigned short minPwdLength;     /* 口令的最小长度 */
        unsigned short maxPwdLength;     /* 口令的最大长度 */
}CDA_PassWordCapability;
typedef struct{
        char *pAlgoNameList;/* 支持的算法的名字的列表 */
        long len;                /* 支持的算法的名字的列表的长度 */
}CDA_AlgorithmCapability;          /* 算法的名字的列表包括一个或多个算法的名字,每一个
算法的名字以'\0'结束。长度是一个或多个算法的名字长度的总和,包括结束字符'\0'。
例如:字符串"des\x00ssf33\x00"     长度     10   */
typedef struct{
        CDA_PassWordCapability pwdInfo;
        CDA_CertificateCapability certInfo;
        CDA_AlgorithmCapability     symmtricInfo;
        CDA_AlgorithmCapability     asymmetricInfo
}C_DeviceCapability;          /* 定义设备的能力 */
typedef struct{
        char *pModeNameList;     /* 加密方式的名字的列表 */
        long lModeListLen;       /* 加密方式的名字的列表的长度 */
        char *pIV;               /* 设置初始化向量的参数 */
        long lIvLen;             /* 初始化向量的字节的长度 */
        char *pPadNameList;      /* 补位方式的名字的列表 */
        long lPadListLen;        /* 补位方式的名字的列表的长度 */
}CDA_AlgorithmInfo;       /* 加密方式的名字的列表包括一个或多个加密方式的名字,每一个
加密方式的名字以'\0'结束。长度是一个或多个加密方式的名字长度的总和,包括结束字符'\0'。
例如:字符串"ecb\x00cbc\x00"          长度     8   */
/* 补位方式的名字的列表包括一个或多个补位方式的名字,每一个补位方式的名字以'\0'结束。
长度是一个或多个补位方式的名字长度的总和,包括结束字符'\0'。
例如:字符串"pad\x00nopad\x00"     长度     10   */
```

C.2.3 函数定义和说明

C_OpenDevice

函数声明:

```
int C_OpenDevice(C_HANDLE *handle,char *szPwd);
```

功能简介:

打开设备,参数 handle 保存打开的设备的句柄。

输入:

要求用户输入打开设备的口令。如果口令为 NULL,返回值为 CDE_SUCCESS。设置 handle 为 NULL_PTR。

输出:

如果打开设备成功,返回值为 CDE_SUCCESS,设置 handle 打开的设备的句柄。

如果打开设备失败,返回值为错误码,设置 handle 为 NULL_PTR。

返回值:

CDE_SUCCESS

CDE_DEVICE
CDE_WRONG_PIN
CDE_WRONG_PARAMS

C_CloseDevice

函数声明：

void C_CloseDevice(C_HANDLE * handle,Char * szPwd);

功能简介：

关闭设备。

输入：

要求用户输入设备的保护口令。

输出：

设置 handle 为 NULL_PTR。

返回值：

无

C_GetDeviceCapability

函数声明：

void C_GetDeviceCapability(CDA_DeviceCapability * pCapability)

功能简介：

获取设备的能力。

输入：

指向结构 CDA_DeviceCapability 的指针。首先初始化一个 CDA_DeviceCapability 结构的实例，将结构的各字段清零。传递结构实例的指针给函数。

输出：

动态库根据自己设备的能力为结构赋值。

返回值：

无

C_GetAlgoInfo

函数声明：

void C_GetAlgoInfo(char * algoName,
CDA_AlgorithmInfo * pInfo)

功能简介：

获取指定算法名字的算法的信息。

输入：

algoName 指定的算法的名字。

PInfo 指向结构 CDA_AlgorithmInfo 的指针。首先初始化一个 CDA_AlgorithmInfo 结构的实例，将结构的各字段清零。传递结构实例的指针给函数。

输出：

动态库参数 pInfo 赋值。

返回值：

无

C_DestroyKeyHandle

函数声明：

int C_DestroyKeyHandle(C_KEY_HANDLE * keyHandle)

功能简介：

销毁密钥。

输入：

如果密钥是一个有效的密钥，释放密钥句柄占用的内存空间，销毁密钥的句柄。

如果密钥是一个无效的密钥，不做任何处理。返回值为 CDE_INVALID_HANDLE。

输出：

如果密钥是一个有效的密钥，释放密钥句柄占用的内存空间，销毁密钥的句柄。设置参数 keyHandle 为 NULL_PTR。

返回值：

CDE_SUCCESS

CDE_INVALID_HANDLE

C_CreatePrivacyKeyPairDir

函数声明：

```
int C_CreatePrivacyKeyPairDir(C_HANDLE   handle,
                CDA_KEYUSAGE   keyUsage,
                Char * szPwd);
```

功能简介：

创建私有的秘密的密钥对目录。

输入：

handle　打开设备的句柄

keyUsage　　密钥的用法

szPwd　　保护此目录的口令

如果目录不存在，创建和指定密钥用法相应的目录，设置目录的保护口令。

如果目录存在，验证口令。如果口令正确，重新创建目录。如果口令不正确，返回值为 CDE_WRONG_PIN

输出：

无

返回值：

CDE_SUCCESS

CDE_CREATE_DIRECTORY

CDE_WRONG_PIN

C_GenerateKeyPair

函数声明：

```
int C_GenerateKeyPair(C_HANDLE   handle,
            char * algoName,
            CDA_KEYUSAGE keyUsage,
            Char * szPwd,
            CDA_PeriodOfValid * period,
```

C_KEY_HANDLE * publicKey,
C_KEY_HANDLE * privateKey);

功能简介：

产生密钥对，指定算法名字、密钥用法、目录保护口令、密钥的有效期。产生密钥对之前必须创建密钥对目录，产生新的公钥和新的私钥，创建公钥和私钥，并且输出公钥和私钥的句柄。如果参数algoName 为 NULL_PTR，动态库产生缺省的密钥对产生算法的密钥对。

输入：

handle　　打开设备的句柄
algoName　　密钥产生算法的名字
keyUsage　　密钥的用法
szPwd　　保护此目录的口令
period　　密钥的有效期

如果和指定的密钥用法相应的目录不存在，返回 CDE_DIRECTORY_NO_EXIST。

验证保护目录的口令，如果口令正确，产生密钥对。如果口令不正确，返回值为 CDE_WRONG_PIN。

输出：

publicKey　　公钥的句柄
privateKey　　私钥的句柄

如果产生密钥对失败，设置 publicKey 和 privateKey 为 NULL_PTR. 。

返回值：

CDE_SUCCESS
CDE_ DIRECTORY_NO_EXIST
CDE_WRONG_PIN
CDE_GENERATE_KEY_PAIR
CDE_MEMORY

C_ExportPublicKey

函数声明：

int C_ExportPublicKey(C_HANDLE　handle,
C_KEY_HANDLE publicKey,
Unsigned char * output,
unsigned long * outputLen);

功能简介：

导出指定的公钥句柄的公钥信息。

输入：

handle　　打开设备的句柄
publicKey　　公钥的句柄

如果输入的公钥的句柄是一个无效的公钥句柄，返回 CDE_INVALID_HANDLE。

输出：

output　　保存公钥信息的缓冲区的地址
outputLen　　保存公钥信息的缓冲区的长度

参数 outputLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的公钥的信息，返回 CDE_SHORT_BUFFER。否则，输出公钥的信息，并且输出公钥信息的长度。

返回值：

CDE_SUCCESS
CDE_ INVALID_HANDLE
CDE_SHORT_BUFFER

C_GetPrivateKeyOfCertificate

函数声明：

```
int C_GetPrivateKeyOfCertificate(C_HANDLE handle,
            char * szPwd,
            CDA_Certificate cert,
            C_KEY_HANDLE * privateKey);
```

功能简介：

获取和参数 cert 指定的证书类型相应的私钥。如果私钥存在。创建这个私钥，并且输出私钥的句柄。否则，返回 CDE_KEY_NO_EXIST。

输入：

handle　　打开设备的句柄
szPwd　　保护密钥目录的口令
cert　　证书的类型

如果私钥存在，验证保护私钥目录的口令，如果口令正确，为这个私钥分配它占用的内存，创建私钥。如果口令不正确，返回值为 CDE_WRONG_PIN。

输出：

privateKey　　私钥的句柄

如果创建私钥失败，设置 privateKey 为 NULL_PTR.。

返回值：

CDE_SUCCESS
CDE_ KEY_NO_EXIST
CDE_WRONG_PIN
CDE_MEMORY

C_GetPublicKeyOfCertificate

函数声明：

```
int C_GetPublicKeyOfCertificate(C_HANDLE handle,
            CDA_Certificate cert,
            C_KEY_HANDLE * publicKey);
```

功能简介：

获取和参数 cert 指定的证书类型相应的公钥。如果公钥存在。创建这个公钥，并且输出公钥的句柄。否则，返回 CDE_KEY_NO_EXIST。

输入：

handle　　打开设备的句柄
cert　　证书的类型

如果私钥存在，为这个公钥分配它占用的内存，创建公钥。

输出：

publicKey　　公钥的句柄

如果创建公钥失败，设置 publicKey 为 NULL_PTR. 。

返回值：

CDE_SUCCESS

CDE_ KEY_NO_EXIST

CDE_MEMORY

C_ImportPrivateKey

函数声明：

```
int C_ImportPrivateKey(C_HANDLE handle,
                       char * algoName,
                       CDA_KEYUSAGE keyUsage,
                       CDA_PeriodOfValid * period,
                       Char * szPwd,
                       C_KEY_HANDLE keyEncryptKey,
                       Unsigned char * cipherPrivateKey,
                       Unsigned long cipherLen);
```

功能简介：

导入和指定的密钥用法相应的私钥。导入私钥之前必须创建密钥对目录。并且必须获取加密私钥的密钥的句柄。如果参数 algoName 为 NULL_PTR，表示加密的私钥的产生算法和动态库缺省的密钥对产生算法一致。

输入：

handle	打开设备的句柄
algoName	密钥产生算法的名字
keyUsage	密钥的用法
szPwd	保护此目录的口令
period	密钥的有效期
keyEncryptKey	解密私钥的密钥的句柄
cipherPrivateKey	私钥的密文
cipherLen	私钥密文的长度

如果和指定的密钥用法相应的目录不存在，返回 CDE_DIRECTORY_NO_EXIST。

验证保护目录的口令，如果口令正确，产生密钥对。如果口令不正确，返回值为 CDE_WRONG_PIN。

如果输入的解密私钥的密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的解密私钥的密钥的密钥用法不能解密，返回 CDE_KEY_USAGE。

如果输入的解密私钥的密钥的过期，返回 CDE_INVALID_PERIOD。

如果解密的过程失败，返回 CDE_DECRYPT。

输出：

无

返回值：

CDE_SUCCESS

CDE_DIRECTORY_NO_EXIST

CDE_WRONG_PIN

CDE_INVALID_HANDLE

CDE_DECRYPT

CDE_KEY_USAGE
CDE_INVALID_PERIOD

C_CreateCertificateDir

函数声明：

int C_CreateCertificateDir(C_HANDLE handle,
CDA_Certificate cert);

功能简介：

创建证书目录。

输入：

handle 打开设备的句柄
cert 证书的类型

如果目录不存在，创建和指定证书的类型相应的目录。

如果目录存在，返回为 CDE_DIRECTORY_EXIST。

输出：

无

返回值：

CDE_SUCCESS
CDE_CREATE_DIRECTORY
CDE_DIRECTORY_EXIST

C_ExportCertificate

函数声明：

int C_ExportCertificate(C_HANDLE handle,
CDA_Certificate cert,
Unsigned char * output,
Unsigned long * outputLen);

功能简介：

导出指定的证书的类型的证书。导出证书之前，指定的证书目录必须存在，并且证书存在。

输入：

handle 打开设备的句柄
cert 证书的类型

如果证书目录不存在，返回 CDE_DIRECTORY_NO_EXIST。

如果证书不存在，返回 CDE_CERTIFICATE_NO_EXIST。

输出：

output 保存证书信息的缓冲区的地址
outputLen 保存证书信息的缓冲区的长度

参数 outputLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的证书的信息，返回 CDE_SHORT_BUFFER。否则，输出证书的信息，并且输出证书信息的长度。

返回值：

CDE_SUCCESS
CDE_SHORT_BUFFER
CDE_DIRECTORY_NO_EXIST

CDE_CERTIFICATE_NO_EXIST

C_ImportCertificate

函数声明：

int C_ImportCertificate(C_HANDLE handle,
CDA_Certificate cert,
Unsigned char * input,
Unsigned long inputLen);

功能简介：

导入指定的证书的类型的证书。导入证书之前，指定的证书目录必须存在。

输入：

handle　打开设备的句柄
cert　证书的类型
input　保存证书信息的缓冲区的地址
inputLen　证书信息的长度

如果证书目录不存在，返回 CDE_DIRECTORY_NO_EXIST。

输出：

无

返回值：

CDE_SUCCESS
CDE_DIRECTORY_NO_EXIST

C_GenerateKey

函数声明：

int C_GenerateKey(C_HANDLE handle,
char * algoName,
CDA_PeriodOfValid * period,
C_KEY_HANDLE * key);

功能简介：

产生和对称密钥产生算法一致的对称密钥。为这个密钥分配它占用的内存，创建密钥，并且输出密钥的句柄。如果参数 algoName 为 NULL_PTR，使用动态库缺省的产生对称密钥的算法。

输入：

handle　打开设备的句柄
algoName　密钥产生的算法
period　密钥的有效期

输出：

key　密钥的句柄

如果产生密钥失败，返回 CDE_INVALID_HANDLE，设置 key 为 NULL_PTR

返回值：

CDE_SUCCESS
CDE_INVALID_HANDLE
CDE_MEMORY

C_ExportKey

函数声明：

```
int C_ExportKey(C_HANDLE handle,
                char * algoName,
                C_KEY_HANDLE key,
                C_KEY_HANDLE keyEncryptKey,
                Unsigned char * cipherKey,
                Unsigned long * cipherKeyLen);
```

功能简介：

使用密钥加密密钥加密密钥的信息，并且输出密文。如果参数 algoName 为 NULL_PTR，使用动态库缺省的对称算法加密密钥。

输入：

handle　　打开设备的句柄

algoName　　加密密钥的算法的名字

key　　被加密的密钥的句柄

eyEncryptKey　　密钥加密密钥的句柄

如果输入的待加密的密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的密钥加密密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的密钥加密密钥的密钥用法不能加密，返回 CDE_KEY_USAGE。

如果输入的密钥加密密钥过期，返回 CDE_INVALID_PERIOD。

如果加密的过程失败，返回 CDE_ENCRYPT。

输出：

cipherKey　　保存密文的缓冲区的地址

cipherKeyLen　　保存密文的缓冲区的长度

参数 cipherKeyLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的密文的信息，返回 CDE_SHORT_BUFFER。否则，输出密文的信息，并且输出密文的长度。

返回值：

CDE_SUCCESS

CDE_INVALID_HANDLE

CDE_SHORT_BUFFER

CDE_ENCRYPT

CDE_KEY_USAGE

CDE_INVALID_PERIOD

C_ImportKey

函数声明：

```
int C_ImportKey(C_HANDLE handle,
                char * algoName,
                CDA_PeriodOfValid * period,
                C_KEY_HANDLE keyDecryptKey,
                Unsigned char * cipherKey,
                Unsigned long cipherKeyLen,
                C_KEY_HANDLE * key);
```

功能简介：

导入和一个对称密钥。解密密钥的密文，创建一个对称的密钥，分配它占用的内存，并且输出这个对称密钥的句柄。导入密钥之前必须获取解密密钥的密钥的句柄。如果参数 algoName 为 NULL_PTR，表示解密密钥算法和动态库缺省的对称加密的算法一致。

输入：

handle　　打开设备的句柄
algoName　　解密密钥算法的名字
period　　密钥的有效期
keyDecryptKey　　解密密钥的密钥的句柄
cipherKey　　密钥的密文
cipherKeyLen　　密钥密文的长度

如果输入的解密密钥的密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。
如果输入的解密密钥的密钥的密钥用法不能解密，返回 CDE_KEY_USAGE。
如果解密密钥的密钥过期，返回 CDE_INVALID_PERIOD。
如果解密的过程失败，返回 CDE_DECRYPT。

输出：

key　　密钥的句柄
如果创建密钥失败，设置 key 为 NULL_PTR

返回值：

CDE_SUCCESS
CDE_INVALID_HANDLE
CDE_DECRYPT
CDE_KEY_USAGE
CDE_INVALID_PERIOD

C_KeyAgreeStep1

函数声明：

```
int C_KeyAgreeStep1(C_HANDLE handle,
                    char * algoName,
                    unsigned char * input,
                    unsigned long inputLen
                    CDA_PeriodOfValid * period,
                    C_KEY_HANDLE * keyAgree);
```

功能简介：

密钥交换算法的第一步，创建一个交换密钥。为这个交换密钥分配它占用的内存，并且输出交换密钥的句柄。如果参数 algoName 为 NULL_PTR，使用动态库缺省的密钥交换算法。

输入：

handle　　打开设备的句柄
algoName　　密钥交换算法的名字
period　　交换密钥的有效期
input　　密钥交换信息
inputLen　　密钥交换信息的长度

输出：

keyAgree　　交换密钥

如果创建交换密钥失败，返回 CDE_INVALID_HANDLE，设置 keyAgree 为 NULL_PTR.。

返回值：

CDE_SUCCESS

CDE_INVALID_HANDLE

C_KeyAgreeStep2

函数声明：

```
int C_KeyAgreeStep2(C_HANDLE handle,
                    char * algoName,
                    C_KEY_HANDLE privateKey,
                    C_KEY_HANDLE keyAgree,
                    CDA_PeriodOfValid * period,
                    C_KEY_HANDLE * shareKey);
```

功能简介：

密钥交换算法的第二步，创建一个共享密钥。为这个共享密钥分配它占用的内存，并且输出共享密钥的句柄。如果参数 algoName 为 NULL_PTR，使用动态库缺省的密钥交换算法。

输入：

handle　　打开设备的句柄

algoName　　密钥交换算法的名字

period　　共享密钥的有效期

privateKey　　私钥的句柄

keyAgree　　交换密钥的句柄

如果输入的私钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的交换密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

输出：

shareKey　　共享密钥的句柄

如果产生共享密钥失败，返回 CDE_INVALID_HANDLE，设置 shareKey 为 NULL_PTR.。

返回值：

CDE_SUCCESS

CDE_INVALID_HANDLE

C_Sign

函数声明：

```
int C_Sign(C_HANDLE  handle,
           char * algoName,
           CDA_AlgorithmInfo * pInfo,
           C_KEY_HANDLE privateKey,
           Unsigned char * input,
           Unsigned long inputLen,
           Unsigned char * signature,
           Unsigned long * signatureLen);
```

功能简介：

计算数字签名，并且输出签名。如果参数 algoName 为 NULL_PTR，使用动态库缺省的数字签名算法。如果参数 pInfo 为 NULL_PTR，使用动态库缺省的数字签名算法的算法参数设置。

输入：

handle	打开设备的句柄
algoName	数字签名算法的名字
pInfo	数字签名算法的算法参数设置
privateKey	私钥的句柄
input	明文
inputLen	明文的长度

如果输入的私钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的私钥的密钥用法不能签名，返回 CDE_KEY_USAGE。

如果输入的私钥的过期，返回 CDE_INVALID_PERIOD。

输出：

signature	保存签名的缓冲区的地址
signatureLen	保存签名的缓冲区的长度

参数 signatureLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的签名，返回 CDE_SHORT_BUFFER。否则，输出签名，并且输出签名的长度。

返回值：

CDE_SUCCESS

CDE_INVALID_HANDLE

CDE_KEY_USAGE

CDE_INVALID_PERIOD

C_Verify

函数声明：

```
int C_Verify (C_HANDLE handle,
              char * algoName,
              CDA_AlgorithmInfo * pInfo,
              C_KEY_HANDLE publicKey,
              Unsigned char * input,
              Unsigned long inputLen,
              Unsigned char * signature,
              Unsigned long signatureLen);
```

功能简介：

验证数字签名。如果参数 algoName 为 NULL_PTR，使用动态库缺省的数字签名算法。如果参数 pInfo 为 NULL_PTR，使用动态库缺省的数字签名算法的算法参数设置。

输入：

handle	打开设备的句柄
algoName	数字签名算法的名字
pInfo	数字签名算法的算法参数设置
publicKey	公钥的句柄
input	明文

inputLen　　明文的长度
signature　　签名
signatureLen　　签名的长度
如果输入的公钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。
如果输入的公钥的密钥用法不能验证签名，返回 CDE_KEY_USAGE。
如果输入的公钥的过期，返回 CDE_INVALID_PERIOD。
如果验证签名失败，返回 CDE_SIGNATURE。
输出：
无
返回值：
CDE_SUCCESS
CDE_INVALID_HANDLE
CDE_KEY_USAGE
CDE_INVALID_PERIOD
CDE_SIGNATURE

C_Encrypt
函数声明：

```
int C_Encrypt(C_HANDLE handle,
              char * algoName,
              CDA_AlgorithmInfo * pInfo,
              C_KEY_HANDLE Key,
              Unsigned char * input,
              Unsigned long inputLen,
              Unsigned char * cipher,
              Unsigned long * cipherLen);
```

功能简介：
加密。如果参数 algoName 为 NULL_PTR，使用动态库缺省的加密算法。如果参数 pInfo 为 NULL_PTR，使用动态库缺省的加密算法的算法参数设置。
输入：
handle　　打开设备的句柄
algoName　　加密算法的名字
pInfo　　加密算法的算法参数设置
Key　　密钥的句柄
input　　明文
inputLen　　明文的长度
如果输入的密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。
如果输入的密钥的密钥用法不能加密，返回 CDE_KEY_USAGE。
如果输入的密钥的过期，返回 CDE_INVALID_PERIOD。
如果加密失败，返回 CDE_ENCRYPT。
输出：
cipher　　保存密文的缓冲区的地址
cipherLen　　保存密文的缓冲区的长度

参数 cipherLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的密文的长度，返回 CDE_SHORT_BUFFER。否则，输出密文，并且输出密文的长度。

返回值：

CDE_SUCCESS
CDE_INVALID_HANDLE
CDE_KEY_USAGE
CDE_INVALID_PERIOD
CDE_ENCRYPT

C_Decrypt

函数声明：

```
int C_Decrypt(C_HANDLE handle,
              char * algoName,
              CDA_AlgorithmInfo * pInfo,
              C_KEY_HANDLE Key,
              Unsigned char * cipher,
              Unsigned long cipherLen,
              Unsigned char * output,
              Unsigned long * outputLen);
```

功能简介：

解密。如果参数 algoName 为 NULL_PTR，使用动态库缺省的解密算法。如果参数 pInfo 为 NULL_PTR，使用动态库缺省的解密算法的算法参数设置。

输入：

handle	打开设备的句柄
algoName	解密算法的名字
pInfo	解密算法的算法参数设置
Key	密钥的句柄
cipher	密文
cipherLen	密文的长度

如果输入的密钥的句柄是一个无效的句柄，返回 CDE_INVALID_HANDLE。

如果输入的密钥的密钥用法不能解密，返回 CDE_KEY_USAGE。

如果输入的密钥的过期，返回 CDE_INVALID_PERIOD。

如果解密失败，返回 CDE_DECRYPT。

输出：

output	保存明文的缓冲区的地址
outputLen	保存明文的缓冲区的长度

参数 outputLen 既是输入参数又是输出参数。如果输入的缓冲区的长度小于要输出的明文的长度，返回 CDE_SHORT_BUFFER。否则，输出明文，并且输出明文的长度。

返回值：

CDE_SUCCESS
CDE_INVALID_HANDLE
CDE_KEY_USAGE
CDE_INVALID_PERIOD

CDE_DECRYPT

C.2.4 返回代码

```
#define CDE_SUCCESS          0
#define CDE_DEVICE           128
#define CDE_WRONG_PIN        129
#define CDE_WRONG_PARAMS     130
#define CDE_INVALID_HANDLE   131
#define CDE_CREATE_DIRECTORY     132
#define CDE_DIRECTORY_NO_EXIST   133
#define CDE_GENERATE_KEY_PAIR    134
#define CDE_MEMORY           135
#define CDE_KEY_NO_EXIST     136
#define CDE_DECRYPT          137
#define CDE_KEY_USAGE        138
#define CDE_INVALID_PERIOD   139
#define CDE_DIRECTORY_EXIST  140
#define CDE_CERTIFICATE_NO_EXIST     141
#define CDE_SIGNATURE        142
#define CDE_ENCRYPT          143
#define CDE_SHORT_BUFFER     144
```

附 录 D
（资料性附录）
证书认证系统网络结构图

D.1 当 RA 采用 C/S 模式时 CA 的网络结构（参见图 D.1）

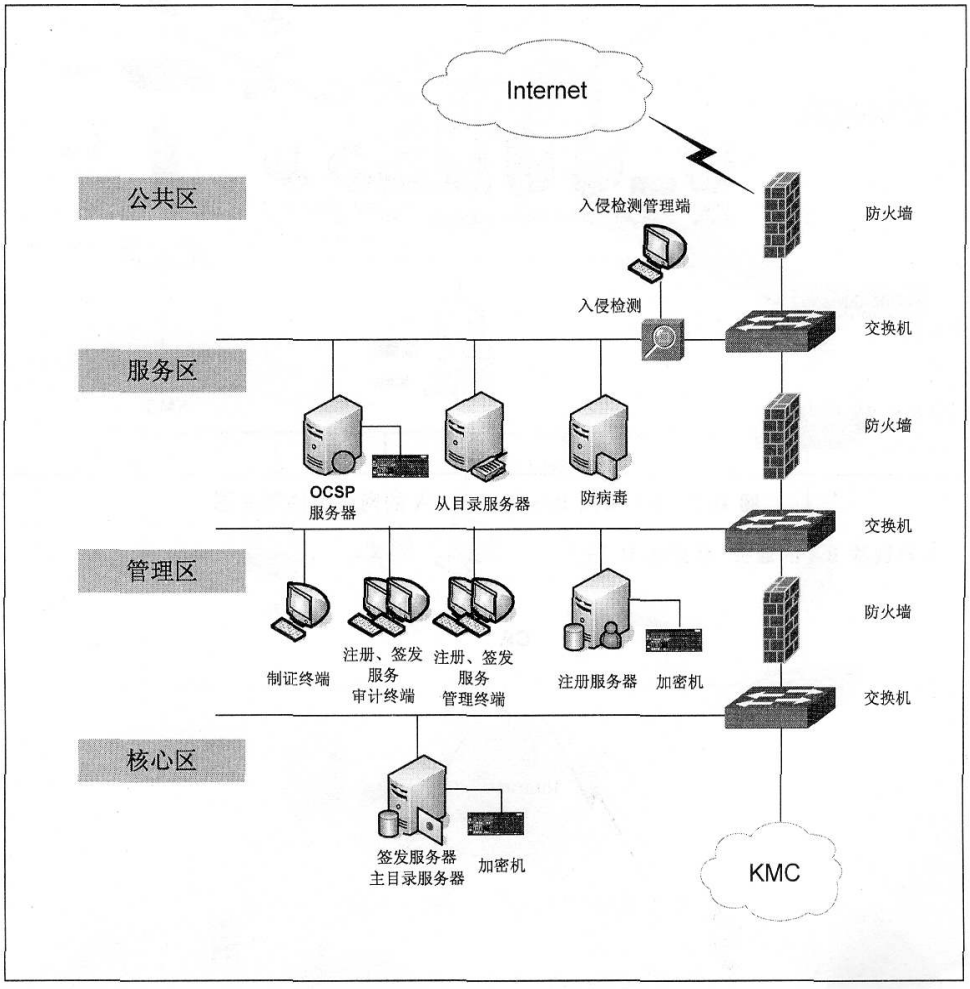

图 D.1 RA 采用 C/S 模式时 CA 的网络结构示意图

D.2　当 RA 采用 B/S 模式时 CA 的网络结构(参见图 D.2)

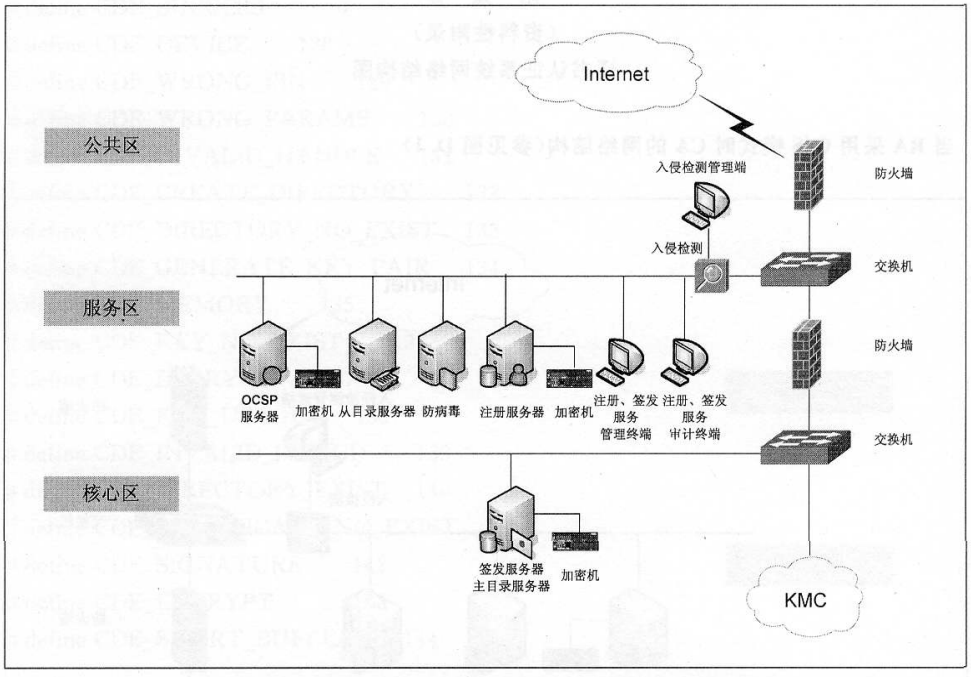

图 D.2　RA 采用 B/S 模式时 CA 的网络结构示意图

D.3　CA 与远程 RA 的连接(参见图 D.3)

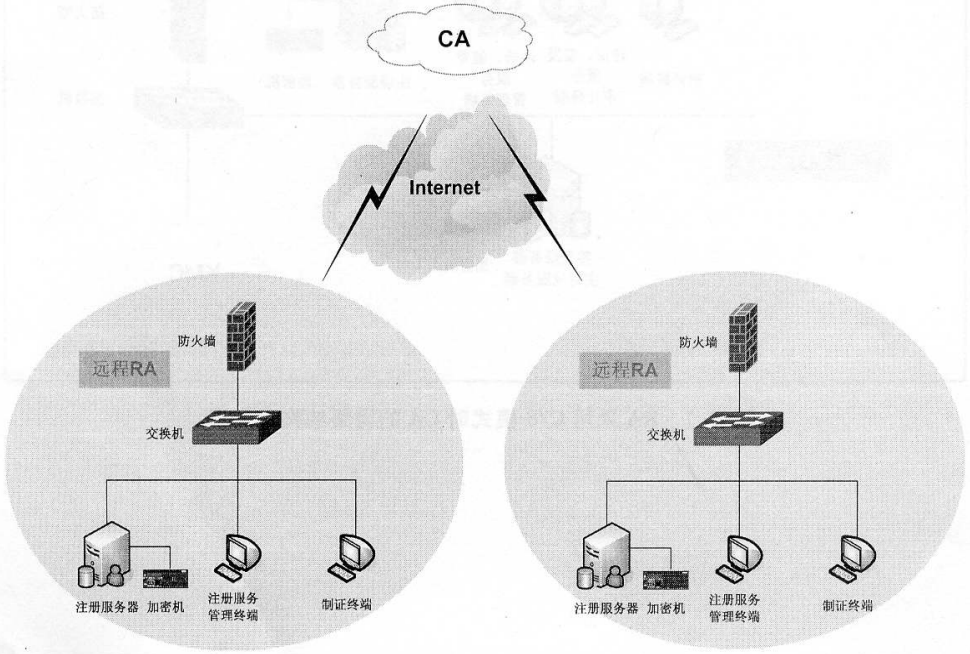

图 D.3　CA 与远程 RA 的连接示意图

D.4 KMC 与多个 CA 的网络连接(参见图 D.4)

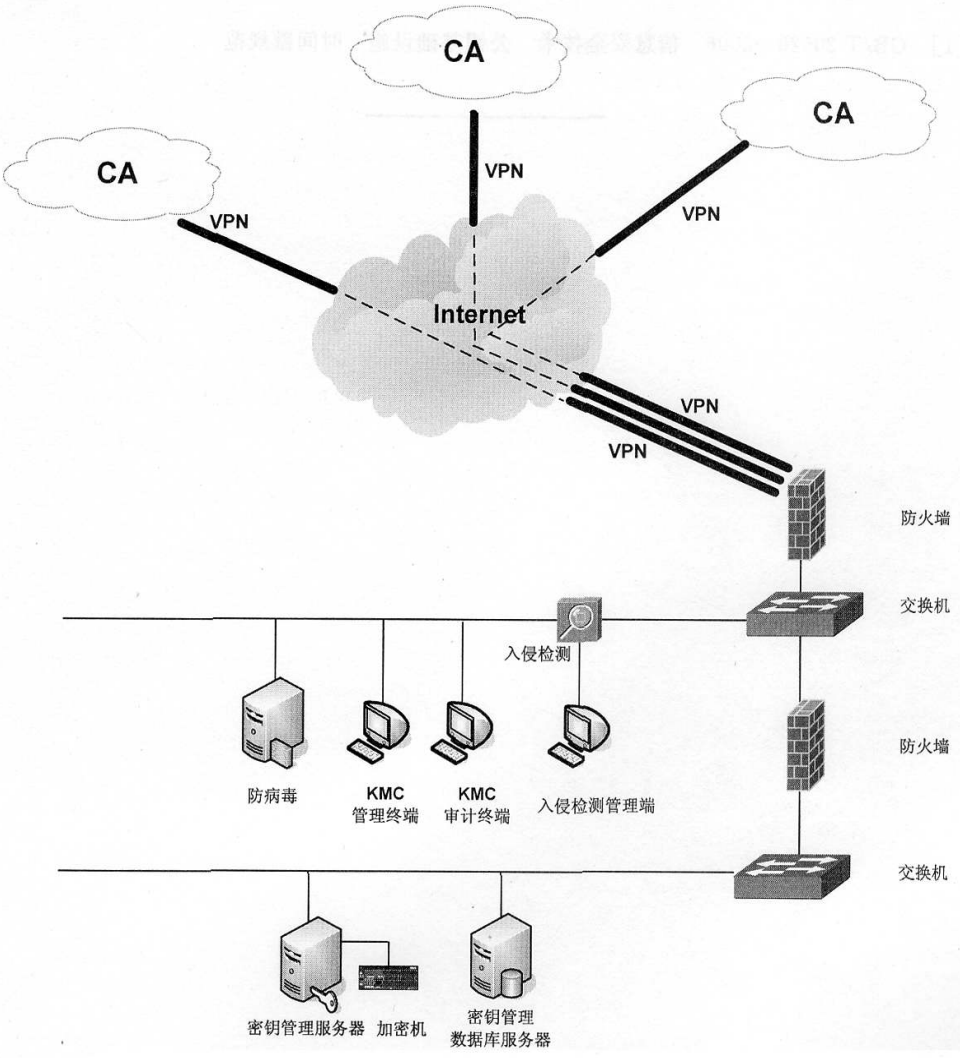

图 D.4 KMC 与多个 CA 的网络连接示意图

参 考 文 献

[1] GB/T 20520—2006 信息安全技术 公钥基础设施 时间戳规范