PCI 密码卡技术规范

Technology Specifications of PCI Cryptography Module

国 家 密 码 管 理 局

2010 年 8 月

目　次

前　言

本规范是国家密码管理局提出并制定的系列规范之一。本规范制定了 PCI 密码卡的技术规范，为信息安全基础设施 PCI 密码卡的研制和开发提供指导和依据。

本规范的附录 A 和附录 B 为规范性附录。附录 A 为 PCI 密码卡 API 函数集，附录 B 为错误代码定义。

本规范由国家密码管理局提出并归口。

本规范起草单位：成都卫士通信息产业股份有限公司、无锡江南计算机技术研究所、兴唐通信科技股份有限公司、济南得安计算机技术有限公司。

本规范主要起草人：霍卫华、徐强、高志权、李玉峰。

本规范责任专家：刘平。

本规范凡涉及密码算法相关内容，按国家有关法规实施。

引　言

本规范的目标是为信息安全基础设施 PCI 密码卡设备制定统一的功能要求、硬件要求、软件要求（含应用接口要求）、安全性要求（含密钥结构）、检测要求等有关内容，通过制定统一的 API 函数集调用 PCI 密码卡设备，向上层提供基础密码服务，实现不同生产厂商的 PCI 密码卡设备互操作，方便应用。为该类密码设备的开发、使用及检测提供标准依据和指导，有利于提高该类密码设备的产品化、标准化和系列化水平。

本规范中涉及的各种密码算法必须获得国家密码管理局审核批准后方可使用。

本规范编制过程中得到了国家商用密码应用技术体系总体工作组的指导。

PCI 密码卡技术规范

1　范围

本规范规定了 PCI 密码卡的功能要求、硬件要求、软件要求（含应用接口要求）、安全性要求（含密钥结构）、检测要求等有关内容。

本规范适用于 PCI 密码卡的研制设计、应用开发，也可用于指导 PCI 密码卡的使用和检测。

2　规范性引用文件

下列标准所包含的条文，通过本规范中的引用而构成为本规范的条文。在标准出版时，所示版本均为有效。考虑到标准的修订，使用本规范时，应研究使用下列标准最新版本的可能性。

	《商用密码管理条例》
GB/T 15277－1994	《分组密码算法工作模式》
GB/T 9813－2000	《微型计算机通用规范》
ISO/IEC 7816-3－1997	《Electronic signals and transmission protocols》
GB/T XXXXX-200X	信息技术 安全技术 密码术语
GB/T XXXXX-200X	公钥密码基础设施应用技术体系 密码设备应用接口规范
GB/T XXXXX-200X	《随机性检测规范》

3　术语和定义

以下术语和定义适用于本规范。

3.1

算法标识 arithmetic identifier

用于标明算法机制的数字化信息。

3.2

大端字节序 big-endian

也称网络字节序，指内存的低地址存放最高有效字节（MSB），即常说的高位在先，低位在后。

3.3

非对称密码算法/公钥密码算法 asymmetric cryptographic algorithm/public key cryptographic algorithm

加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。

3.4

解密 decipherment/decryption

与一个可逆的加密过程相对应的反向运算过程。该过程使用适当的密钥，将已加密的文本转换恢复成明文。

3.5

设备密钥对 device key pair

存储在设备内部的用于设备管理的非对称密钥对，包含签名密钥对和加密密钥对。

3.6

数字签名 digital signature

把输入的消息串使用指定私有密钥经过非对称密码变换为固定长输出串的算法，附加在原始消息上。该算法用于确认数据的来源及其完整性，防止被他人伪造或者签发者否认。

3.7

加密 encipherment/encryption

对数据通过密码算法和适当的密钥进行运算变换来产生密文，以便隐藏数据的信息内容。

3.8

杂凑算法 hash algorithm

将一个任意长的比特串映射到一个固定长的比特串的函数，且满足下列两个特性：

1) 为一个给定的输出找出能映射到该输出的一个输入是计算不可行的；

2) 为一个给定的输入找出能映射到同一个输出的另一个输入是计算不可行的。

3.9

密钥加密密钥 key encrypting key

又称二级密钥(Secondary Key)或密钥传送密钥(key Transport key)，存储在硬件设备内部，用于加密保护会话密钥的对称密钥。

3.10

小端字节序 little-endian

指内存的低地址存放最低有效字节（LSB），即常说的低位在先，高位在后。

3.11

消息认证码算法 message authentication code algorithm

又称 MAC 运算算法，把输入的消息串和秘密密钥变化成固定长输出串的算法，输出串称为消息认证码。该算法用于进行消息的完整性和正确性校验。

3.12

PCI 密码卡 PCI cryptography module

指具有密码运算功能和自身安全保护功能的 PCI 硬件板卡设备。

3.13

PCI Express

PCI Express 是 PCI 局域总线的改进版，是一种高性能的局域 I/O 总线互连技术，保持 PCI 软件的使用模型并使用具有多通路高速的串行总线替代物理总线。

3.14

PCI 局部总线 PCI local bus

PCI 是 Peripheral Component Interconnect 的英文缩写，其含义为外设部件互连。PCI 局部总线是和外设部件相连接的一种结构化总线，其特点是：传输带宽高、兼容性强、扩展性好。

3.15

私有密钥 private key

一个实体的非对称密码算法中规定只能由该实体使用不能公开的密钥，该密钥只能由该实体安全存储。

3.16

私钥访问控制码 private key access password

用于获取私钥使用权限的口令字。

3.17

公开密钥 public key

一个实体的非对称密码算法中规定可以公开的密钥。

3.18

随机数 random number

由物理噪声源芯片产生的高质量随机数，该数据是在理论上没有规律可循，不可计算、不可预测、不可重复的时变参数。

3.19

会话密钥 session key

密钥管理中的最低一层密钥，该密钥只能在一次会话中使用，使用结束就销毁。

3.20

对称密码算法 symmetric cryptographic algorithm

原发者和接收者均采用同一秘密密钥进行变换的密码技术(体制)。其中，加密密钥等于解密密钥，或者一个密钥可以从另一个密钥导出的密码体制。

3.21

用户密钥对 user key pair

存储在设备内部的用于应用密码运算的非对称密钥对，包含签名密钥对和加密密钥对。

4 符号和缩略语

下列缩略语适用于本部分：

PCI	外设部件互连，是一种密码设备的硬件实体
PCI Express	PCI 局域总线的改进版，是一种高性能的局域 I/O 总线互连技术
IC	集成电路
DMA	直接存储器访问
API	应用编程接口
RSA	一种公开的密钥密码体制，它由三个发明者名字的首字母缩写
ECC	椭圆曲线算法（Elliptic Curve Cryptography）
IPK	内部加密密钥对公开密钥（Internal Public Key）
ISK	内部加密密钥对私有密钥（Internal Private Key）
EPK	外部加密密钥对公开密钥（External Public Key）
KEK	密钥加密密钥（Key Encrypting Key）
ECB	电码密本链接模式（Electronic Code Book）
CBC	密码分组链接模式（Cipher Block Chaining）
CFB	密文反馈链接模式（Ciphertext Feedback）
OFB	输出反馈链接模式（Output Feedback）
MAC	消息认证码算法
Hash	杂凑算法，又称散列函数算法

5 PCI 密码卡概述

PCI 密码卡是以 PCI 局部总线或者 PCI Express 为接口，具有密码运算功能、密钥管理功能、物理随机数产生功能和设备自身安全保护措施的密码设备。

PCI 密码卡可以应用在需要密码运算和密钥管理等安全功能的、具有 PCI 局部总线或者 PCI Express 的通信设备、计算机设备、安全保密设备上，例如：虚拟专网（VPN）设备、证书中心（CA）系统的有关设备、网络密码机、安全服务器、安全终端、安全管理中心、密钥管理设备等。

6 功能要求

在本规范中，PCI 密码卡的功能包括两类，分别是基本功能和扩展功能：

1) 基本功能，主要包括：a.密码运算功能，包括对称密钥密码算法、非对称密钥密码算法和杂凑算法；b.密钥管理功能；c.两级密钥结构体制（用户密钥对或密钥加密密钥、对称密码算法的会话密钥）的支持功能；d.真随机数生成和检验功能；e.密码卡内部敏感数据信息的安全保护功能等。
2) 扩展功能，是在基本功能的基础上，根据应用需要进行合理的扩充和增添的功能，包括 a.对密码卡的访问控制功能，如必须通过口令验证才能使用密码运算功能；b.非对称密钥密码算法可选择支持 RSA 算法或者 ECC 算法中的任意一种；b.三级密钥结构体制的支持功能（在两级密钥结构体制的基础上引入设备保护密钥）；c.除了必须支持的对称密码算法、非对称密码算法和杂凑算法之外的其他密码运算功能等；d.密码卡内部的文件存储管理功能。

6.1 密码运算功能

PCI 密码卡必须支持对称密钥密码算法、非对称密钥密码算法和杂凑算法，在适当的驱动条件下完成相应的密码算法运算。根据实际需要，PCI 密码卡选择支持其他密码算法。

6.1.1 对称密钥密码算法

PCI 密码卡必须支持至少一种对称密钥密码算法，如分组密码算法、序列密码算法。

数据加解密算法、敏感数据信息保护算法、MAC 算法均可采用对称密钥密码算法来实现。

6.1.2 分组密码算法的链接模式

PCI 密码卡提供的分组密码算法必须支持 ECB 和 CBC 两种链接模式，其他链接模式（例如 CFB 或者 OFB）作为扩展功能选择支持。

分组长度大于 64 比特的分组密码的工作方式的实现应与 GB/T 15277－1994 《信息处理 64bit 分组密码算法的工作方式》中的描述相一致。

6.1.3 杂凑算法

PCI 密码卡必须支持至少一种杂凑算法。

6.1.4 非对称密钥密码算法

数字签名、签名验证、密钥安全存储管理等可采用非对称密钥密码算法来实现。对非对称密钥密码算法的基本要求为：

1) 非对称密钥密码算法必须支持至少一种非对称密钥密码算法；算法运算可以采用硬件或软件方法来实现；
2) 选择支持 RSA 算法时，应具备 1024 比特及以上的模长；
3) 选择支持 ECC 算法时，应具备 256 比特及以上的模长；
4) 应提供数字签名和签名验证功能；
5) 选择支持 ECC 算法时，PCI 密码卡应提供 ECC 密钥协商生成会话密钥功能。

6.1.5 扩展密码算法

PCI 密码卡只在需要时才选择支持扩展密码算法。

6.2 密钥管理功能

PCI 密码卡必须具有 IC 卡接口或者 USB 接口。通过将 PCI 密码卡、IC 卡或者 USBKey

两部分硬件实体结合起来，完成用户身份认证功能。只有通过用户身份认证的用户方可使用PCI 密码卡提供的密钥管理服务。IC 卡或者 USBKey 可以作为公开密钥、私有密钥、密钥加密密钥、用户签名信息、证书信息或其他敏感数据信息的载体。

IC 卡的类型为智能 IC 卡；配用的智能 IC 卡或者 USBKey 必须通过国家密码管理局的审核批准。

6.2.1 密钥的种类和作用

PCI 密码卡必须支持至少两级密钥结构体制：一级是用户密钥对或者密钥加密密钥，一级是会话密钥。这种密钥结构体制支持基于 PCI 密码卡的设备或应用系统实现端到端的数据安全保护、一次一密以及相应的密钥管理等安全服务。其中，用户密钥对或者密钥加密密钥选择支持任意一种，也可选择支持两种。

此外，作为硬件设备自身的维护管理，选择支持设备密钥对。

1) 设备密钥对

分加密密钥对和签名密钥对两种，选择支持 RSA 算法或者 ECC 算法的任意一种；用于实现数字签名、签名验证等功能；其中，私有密钥必须受私钥访问控制码的安全访问控制。

2) 用户密钥对

分加密密钥对和签名密钥对两种，选择支持 RSA 算法或者 ECC 算法的任意一种；用于实现用户数字签名、签名验证以及会话密钥的安全保护等功能；其中，私有密钥必须受私钥访问控制码的安全访问控制。

3) 密钥加密密钥

长度不低于 128 比特，用于实现会话密钥的安全保护功能。

4) 会话密钥

长度不低于 128 比特，用于用户数据加解密运算，或者作为其他应用系统密钥的加密保护密钥。

在上述两级密钥结构体制的基础上，选择支持增加第三级密钥，即保护密钥，作为用户密钥对或者数据加密密钥的安全存储保护密钥。

6.2.2 密钥的产生及存储

1) 设备密钥对

由设备初始化时使用的管理工具生成或者安装，存储在 PCI 密码卡硬件内部。

2) 用户密钥对

由密码设备管理工具生成或者安装，必须支持使用物理噪声源芯片生成，必须支持使用强素数；根据系统需要必须支持一定数量用户密钥对的存储区域；用户密钥对的私有密钥必须支持硬件内部安全存储，必须支持私钥访问控制码的安全访问控制。

3) 密钥加密密钥

由密码设备管理工具生成或者安装，必须支持物理噪声源芯片生成；根据系统需要必须支持一定数量密钥加密密钥的存储区域；该密钥必须支持硬件内部安全存储。

4) 会话密钥

必须支持使用物理噪声源芯片生成，以确保会话密钥的质量；必须支持一次会话更换一次会话密钥；不得以明文方式导出硬件；在会话密钥长期存储时，必须支持用户密钥对或者密钥加密密钥加密存储等安全保护措施。

此外，对于支持非对称密钥密码算法运算的 PCI 密码卡，必须支持非对称密钥对的生成功能，必须支持使用物理噪声源芯片生成，必须支持使用强素数。

6.2.3 密钥的使用及更换

1) 设备密钥对

定期更换不作具体规定。

2) 用户密钥对

用于用户数字签名、签名验证运算，或者作为会话密钥的加密保护密钥对；定期更换由密码设备管理工具完成，这里不作具体规定。

3) 密钥加密密钥

作为会话密钥的加密保护密钥;定期更换由密码设备管理工具完成,这里不作具体规定。

4) 会话密钥

用于用户数据加解密运算，或者作为其他应用系统密钥的加密保护密钥；在会话密钥用于用户数据的加解密运算时，必须支持经常更换会话密钥，以确保用户数据加解密运算的安全性；每次会话中使用的会话密钥不相同，必须支持一次会话更换一次会话密钥。

6.2.4 密钥的销毁

1) 设备密钥对

密钥销毁不作具体规定。

2) 用户密钥对

应提供紧急销毁的安全保护措施，以确保 PCI 密码卡自身的安全性。

3) 密钥加密密钥

应提供紧急销毁的安全保护措施，以确保 PCI 密码卡自身的安全性。

4) 会话密钥

必须支持，以确保用户数据加解密运算的安全性。

6.2.5 密钥的备份及恢复

1) 设备密钥对

由设备初始化时使用的管理工具完成，这里不作具体规定。

2) 用户密钥对

由密码设备管理工具完成，这里不作具体规定。

3) 密钥加密密钥

由密码设备管理工具完成，这里不作具体规定。

4) 会话密钥

由上层应用系统进行统一管理，这里不作具体规定。

6.3 随机数生成和检验功能

PCI 密码卡内部必须支持物理噪声源芯片，提供真随机数序列的生成。PCI 密码卡必须支持对物理噪声源芯片产生的随机数进行统计检验，以保证其质量。随机数检验的质量标准参照国家密码管理局颁布的《随机性检测规范》。

6.4 设备标志信息存储功能

PCI 密码卡应具有唯一的设备标志信息，以区分不同厂家、不同型号的 PCI 密码卡。设备标志信息包括以下几个部分：生产厂商、设备类型、序列号、硬件版本、软件版本、支持算法等。国家密码管理局和用户可以通过这些信息来查验和辨识 PCI 密码卡。设备标志信息通过附录 A.4.1 中的函数 SDF_GetDeviceInfo 获取。

7 硬件要求

7.1 PCI 接口要求

7.1.1 PCI 接口规范

符合 PCI 2.0 规范或者 PCI Express 1.1 规范；考虑到总线规范的不断扩展修订，使用本规范时，应研究使用最新版本规范的可能性。

7.1.2 数据总线

采用PCI局域总线的PCI密码卡,其总线采用32位或者64位数据总线;采用PCI Express规范的PCI密码卡，其总线采用多通路高速的串行总线；采用存储器方式或I/O端口方式进行访问。

基于x86及其兼容平台的PCI密码卡，总线排序方式采用Little-Endian模式；基于PowerPC及其兼容平台的PCI密码卡，总线排序方式采用Big-Endian模式。

7.1.3 通信方式

数据通信可采用查询方式或中断通信方式。

7.1.4 数据传输

采用DMA传输方式或其他方式。

7.1.5 电源和功耗

采用PCI局域总线的PCI密码卡可以工作于单一的“＋5V”电压、单一“＋3.3V”电压或两种混合电压环境；采用PCI Express规范的PCI密码卡可工作于“＋3.3V”和“＋12V”电压环境。在上述任何一种情形下，PCI密码卡的功耗不得大于25W。

7.2 硬件组成要求

PCI密码卡的基本硬件组成包括以下几种部件：密码算法模块、处理器（DSP或CPU)、物理噪声源、非易失性存储器、IC卡接口或者USB接口、PCI桥接芯片或者PCI Express桥接芯片等。

在具体实现中，以上部件不一定独立存在于PCI密码卡中，但上述部件所具有的功能是必不可少的。随着技术的不断发展，不排除多个部件集成于一体的可能性。

IC卡接口应该遵循ISO/IEC 7816-3－1997 《Electronic signals and transmission protocols》中的有关规定。

7.3 环境要求

PCI密码卡的工作环境必须符合GB/T 9813－2000 《微型计算机通用规范》中关于“气候环境适应性”二级的规定要求。关于贮藏运输温度、工作温度、贮藏运输相对湿度和工作相对湿度的环境。

7.3.1 贮藏运输温度

-40℃～55℃。

7.3.2 工作温度

0℃～40℃。

7.3.3 贮藏运输相对湿度

20%～93%（40℃）。

7.3.4 工作相对湿度

30%～90%。

7.4 可靠性要求

PCI密码卡的可靠性指标采用平均无故障工作时间（MTBF）来衡量。PCI密码卡的平均无故障工作时间由组成密码卡的各个部件的可靠性决定。

PCI密码卡的平均无故障工作时间MTBF必须大于10,000小时。

7.5 电路原理图和印制版图要求

PCI密码卡电路原理图、方框图和印制版图的设计和标注应准确清晰。

8 软件要求

PCI密码卡软件设计采用分层设计的方法。PCI密码卡的软件分为三个层次：底层软件

（监控软件）、驱动程序和应用编程接口。为了保证软件部分的安全性和可扩展性，本节对各个层次的实现提出原则性的约束。

8.1 底层软件

8.1.1 模块化设计

底层软件采用模块化设计，以保证不同版本之间模块的向后兼容性。

8.1.2 随机数检验

由物理噪声源芯片产生的随机序列应至少通过几项基本的随机性（如 0 和 1 的平衡性、连长特性等）检验，以确保其等概性和不可预测性，随机数检验的质量标准参照国家密码管理局颁布的《随机性检测规范》。只有通过检验的随机序列才可用作随机密钥。

8.1.3 其他要求

底层软件应通过技术措施，防止用户的非法调用，防止系统“死机”。

8.2 驱动程序

8.2.1 透明传输

驱动程序应透明传输应用系统和 PCI 密码卡缓存区之间的数据，不得截获、解析应用系统的数据结构。

8.2.2 安装/卸载

应支持安装/卸载驱动程序。

8.2.3 多设备支持要求

驱动程序应该支持多个 PCI 密码卡设备同时使用和操作的基本要求。

8.3 应用编程接口（API）

8.3.1 字节顺序

在不同型号的 PCI 密码卡之间加解密互通传输时，为避免因字节顺序差异带来的影响或错误，这里定义数据的存储顺序和传输顺序按照大端字节序（big-endian）进行处理。

下面举例说明，数据 0x12345678，其对应的大端字节序存储顺序如图 1 所示。

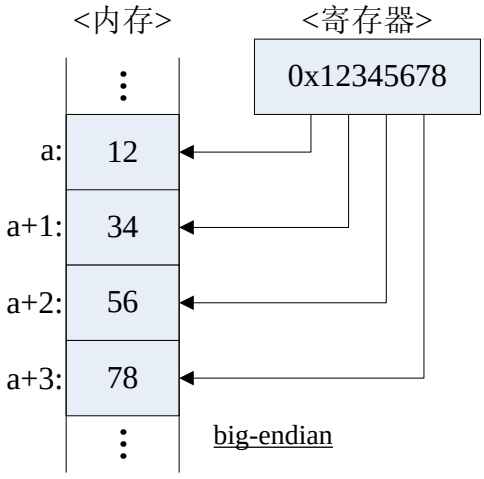

图 1 大端字节序存储示意图

8.3.2 对称分组密码算法数据填充

在使用对称密钥分组密码算法对数据加密时，当实际的数据长度不是分组字节长度的整数倍时，首先进行冗余字节的填充，使数据长度按照分组长度进行边界对齐后，再做加密处理；解密后，应按照填充方式，删除解密结果中填充的冗余字节。

数据填充的方法必须按照 PKCS#5 填充方式进行数据填充，即补码值为需要补码的位数。

PKCS#5 的填充格式为：DATA ‖ PADDING

其中，DATA 为用户数据；PADDING 为填充值。设 n 为分组的字节长度，DATALEN 为用户

数据的长度，则需要补码的位数 P 取值为：

P = DATALEN mod n

如果 P 不等于 0，则在用户数据后面添加 P 个字节的数 P；反之则在用户数据后面添加 n 个字节的数 n。

8.3.3 非对称密码算法数据填充

在使用 RSA 非对称密码算法对数据加密时，必须按照 PKCS#1 填充方式进行数据填充。

PKCS#1 的填充格式为：00 | BT | PADDING | 00 | DATA

其中，BT 为数据块类型；PADDING 为填充值，DATA 为用户数据。

BT 为 1 时表示该数据块为签名数据块，后续填充值为 0xFF；BT 为 2 时表示该数据块为加密数据块，后续填充值为非 0 的随机数。

PADDING 填充值的长度为：K - 3 - DATALEN

其中，K 为 RSA 模长的字节数；DATALEN 为用户数据的长度，要求长度小于 K－3－8 字节。

8.3.4 应用编程接口（API）的目标模块

应用编程接口（API）以目标模块方式提供给用户，作为 PCI 密码卡用户应用系统和驱动程序之间的接口。目标模块可以是动态库、静态库或其他二进制目标代码形式，不经主管部门批准不可将源代码直接提供给用户。PCI 密码卡生产厂家必须对应用编程接口给出详细具体的使用说明，不得泄漏密码算法的细节和与使用无关的密钥管理的细节。

应用编程接口规范应遵守《公钥密码基础设施应用技术体系 密码设备应用接口规范》中的有关规定。

8.3.5 多个私钥的支持

在 PCI 密码卡硬件内部选择支持多个公私钥对，应用编程接口必须支持对多个公私钥对的使用和操作。

8.3.6 API 函数集

PCI 密码卡的 API 函数集定义请参见附录 A。

9 安全性要求

9.1 密码算法安全

必须支持使用经国家密码管理局审查批准的密码算法和密码模块芯片。

9.2 密钥管理安全

基于本规范设计、开发的 PCI 密码卡在密钥管理方面，应满足以下要求：

1) 密码卡生成的密钥或密钥对必须支持使用物理噪声源芯片硬件产生，并通过随机性检验；
2) 在任何时间、任何情况下，PCI 密码卡生成的会话密钥均不能以明文形式出现在硬件设备外部；
3) 在任何时间、任何情况下，硬件存储用户密钥对的私有密钥均不能以明文形式出现在硬件设备外部；
4) 在任何时间、任何情况下，硬件存储的密钥加密密钥均不能以明文形式出现在硬件设备外部；
5) 硬件存储用户密钥对的私有密钥必须支持私钥访问控制码的安全访问控制，防止非法使用和导出；
6) 硬件存储的密钥应具备有效的密钥保护机制，防止解剖、探测和非法读取；
7) 硬件存储的用户密钥对或者密钥加密密钥的定期更换、备份及恢复由密码设备管理

工具完成，这里不作具体规定。

8) PCI 密码卡必须支持有效的安全机制和措施，保证密钥在生成、安装、导入、存储、备份、恢复及销毁整个生存期间的安全，此安全机制可由设备厂商自行设计实现。

9.3 随机数安全

必须支持使用经国家密码管理局审查批准的物理噪声源芯片作为真随机数生成器。

9.4 设备管理安全

PCI 密码卡必须支持具有足够强的设备自身安全保护措施，防止非法用户的非法使用。

其中，硬件存储用户密钥对的私有密钥应具备私钥访问控制码的控制机制，防止非法使用和导出。硬件存储用户密钥对的私有密钥，其访问控制码的设置应由密码设备管理工具完成，可采用口令字方式，口令字编码长度应不低于 8 字符，同时口令字内容应为字符与数字的混合体。

此外，作为硬件设备自身的维护管理，选择支持支持设备密钥对。

9.5 硬件安全

1) 物理保护

PCI 密码卡作为一个软硬件相结合的系统，必须支持具有足够强的物理保护措施，防止底层代码被监听、拷贝、剖析、跟踪，确保 PCI 密码卡内密码算法和密钥的安全。重要安全参数的保护允许反熔丝 FPGA、带保护算法但无反熔丝两种方式，PCI 密码卡的安全性设计方案必须通过国家密码管理局审查批准。

2) 敏感数据信息的安全存储保护

PCI 密码卡的敏感数据信息包括：存放在 PCI 密码卡及附件（如 IC 卡或者 USBKey）内部的算法程序、密钥、随机因子、密码算法因子及其它不能公开的用户身份信息以及与实体有关的认证数据等。PCI 密码卡硬件存储的这些数据必须支持安全存储保护，以确保非法用户、剽窃者和偷袭者无法获取这些敏感数据信息。

在 PCI 密码卡正常工作过程中，敏感数据信息不以明文形式出现在硬件设备外部。

9.6 软件安全

1) 底层软件

底层软件应采用模块化设计，以保证不同版本之间模块的向后兼容性。底层软件应通过技术防范措施，防止用户的非法调用，防止系统“死机”。

2) 驱动程序

驱动程序必须支持透明传输上层应用系统和 PCI 密码卡硬件设备缓存区之间的数据，不得截获、解析应用系统中的数据结构。

3) 应用编程接口

应用编程接口应遵守《公钥密码基础设施应用技术体系 密码设备应用接口规范》中的有关规定，能够防止非法用户的非法调用，对硬件存储的私有密钥必须支持提供私钥权访问控制码的安全访问控制。

9.7 使用安全

基于本规范设计、开发的 PCI 密码卡在使用方面，应满足以下要求：

1) PCI 密码卡必须支持初始状态和就绪两个状态；
2) 未安装用户密钥对或密钥加密密钥的 PCI 密码卡处于初始状态，已安装用户密钥对或密钥加密密钥的 PCI 密码卡处于就绪状态；
3) 在初始状态下，除可读取硬件设备信息操作外，不能执行任何安全服务操作，在使用密码设备管理工具完成生成（或恢复）用户密钥对或者密钥加密密钥后，PCI 密码卡就处于就绪状态；
4) 在就绪状态下，能执行与 PCI 密码卡硬件设备相关的安全服务操作；

5) 在就绪状态下，进行硬件存储用户密钥对的私有密钥操作前，必须通过私钥访问控制码的安全认证。

10 检测要求

生产厂商研制生产的 PCI 密码卡在投入使用前必须按照本规范规定的检测要求进行各项检测，检测要求规定了 PCI 密码卡的硬件检测、功能检测、性能检测、算法正确性检测和安全性检测等内容。

10.1 硬件检测

10.1.1 安装检测

PCI 密码卡能正确地安装到系统的对应插槽中，其设备驱动程序在指定的操作系统中能够正确地安装和卸载。

10.1.2 初始化测试

PCI 密码卡出厂时处于初始状态，通过使用密码设备管理工具完成生成（或恢复）用户密钥对或者密钥加密密钥后，PCI 密码卡能正确完成初始化操作，处于就绪状态。

10.1.3 硬件上电自检

PCI 密码卡在开机上电后必须支持进行硬件自检，自检结果可以用硬件的方式给出指示；未通过上电自检，PCI 密码卡必须拒绝一切密码功能调用服务。上电自检必须完成以下项目：

1) 物理噪声源是否失效。采集物理噪声源芯片产生的随机数，然后进行随机性检测；
2) 密码运算单元是否失效。对特定的数据进行密码运算，检查计算结果与预知结果是否一致；
3) 静态存储数据的完整性是否被破坏。对静态存储数据计算数据的杂凑值，将计算结果与预知结果作比较；
4) 其它需要进行自检的功能部件是否工作正常。

10.2 功能检测

PCI 密码卡的功能检测主要通过附录 A 统一制定的 PCI 密码卡 API 函数集进行访问检测。这部分测试程序由检测机构提供。API 函数接口库由 PCI 密码卡的研制单位提供。对于正确的调用环境和调用过程，调用 API 函数接口应该返回正确的结果，并完成相应功能；对于设定的不正确的调用环境和调用过程，调用 API 函数接口应返回相应的错误代码。

10.2.1 密码运算功能检测

测试程序对 PCI 密码卡支持的对称密码算法、非对称密码算法和杂凑算法进行基本运算检测，检测其运算的功能正确性。

10.2.2 密钥管理检测

测试程序对于具有对称密码算法密钥管理功能和非对称算法密钥管理功能的 PCI 密码卡，通过制定测试项依次检测其生成密钥、保护导入密钥、保护导出密钥以及销毁密钥等功能项进行功能正确性检测。

10.2.3 物理随机数检测

测试程序通过调用 API 函数接口，PCI 密码卡上的物理噪声源芯片生成并输出指定长度的比特流作为测试样本。将测试样本输入测试程序进行随机性统计特性测试。

10.2.4 密码卡内敏感数据的安全保护检测

对硬件存储和使用敏感数据的实现正确性进行检测，并进行相关的抗攻击性实验。

10.3 性能检测

目的是测试 PCI 密码卡进行各项密码运算的速度指标。每次测试应具有足够的数据量，

保证 PCI 密码卡进入稳定工作期，防止性能波动。本节没有规定对 PCI 密码卡进行速度性能检测的具体操作系统。对于特殊的操作系统，应进行单独测试。

下列各项速度性能测试中的测试量由数据报文长度和测试次数决定。可以根据各个测试项的具体耗时情况，依照等比序列来选取测试次数，例如：测试次数 N 可以选择 1 次、10 次、100 次、1000 次…等，分别测试后得到不同测试次数时的性能序列。数据报文长度的选择在各个速度性能测试项中分别定义。

在 10.3.1、10.3.3 和 10.3.4 中包含的各个测试项的速度性能的计算如下式所示：

$$S = 8LN/(1024\times1024T)$$

其中，S为速度，单位为 Mbps(兆比特/秒)；L为数据报文的长度，单位为字节；N为测试次数；T为测量所耗费的时间，单位为秒。

在 10.3.2 中包含的各个测试项的速度性能的计算如下式所示：

$$S = LN/T$$

其中，S为速度，单位为 tps（次/秒）；L为数据报文的长度（必须是算法模长 M的整数倍），单位为字节；N为测试次数；T为测量所耗费的时间，单位为秒。

10.3.1 对称密码算法的加解密性能测试

将内存中的一个定长数据报文，发送给 PCI 密码卡进行对称密码算法的加密和解密操作，重复操作 N次，测量其完成时间 T。用于测试的数据由检测机构选取。测试应进行多次，结果取平均值。如 PCI 密码卡支持对称算法的多种工作模式，必须测试所支持的各种工作模式，如：ECB、CBC、CFB、OFB 等都进行测试。并且应对所支持的所有使用方式（如加密、解密等）进行逐一测试。

取不同长度的数据报文分别进行测试，并记录结果。下列典型长度的数据报文必须进行测试：一个分组长度、64 字节、128 字节、256 字节、512 字节、1024 字节、2048 字节、10240 字节以及 PCI 密码卡研制单位建议的长度。

10.3.2 非对称密码算法的运算性能测试

将内存中的一个定长数据报文，发送给 PCI 密码卡进行非对称密码算法的运算操作，重复操作 N次，测量其完成时间 T。用于测试的数据由检测机构选取。测试应进行多次，结果取平均值。如 PCI 密码卡支持多种非对称算法，必须测试所支持的所有非对称密钥密码算法及其各种应用模式，如：RSA、ECC 等算法的加密、解密、签名、验证等。

10.3.3 杂凑算法性能测试

将内存中的一个定长数据报文，发送给 PCI 密码卡进行杂凑运算，重复操作 N次，测量其完成时间 T。用于测试的数据由检测机构选取。测试应进行多次，结果取平均值。

取不同长度的数据报文分别进行测试，并记录结果。下列典型长度的数据报文必须进行测试：一个分组长度、64 字节、128 字节、256 字节、512 字节、1024 字节、2048 字节、10240 字节以及 PCI 密码卡研制单位建议的长度。

10.3.4 非对称密钥对生成性能测试

让 PCI 密码卡生成并输出指定数量的密钥对，测量其完成时间。测试应进行多次，结果取平均值。

10.4 算法正确性检测

通过测试程序，使用 PCI 密码卡对已知相应正确结果的数据执行密码运算，然后比较计算结果和预知结果。如果计算结果和预知结果相同，则测试通过；否则，测试失败。密码算法实现正确性测试必须测试 PCI 密码卡提供的每个对称和非对称密钥密码算法的每个功能函数，如：加密、解密、杂凑、签名、验证等。如 PCI 密码卡支持算法的多种工作模式，必须测试所有支持的工作模式，如：ECB、CBC、CFB、OFB 等。

此外，对于具有 ECC 运算及其密钥管理的 PCI 卡，通过测试程序，与其他生产厂商研制

的 PCI 密码卡进行 ECC 密钥协商互通测试，检测其正确性。

10.5 安全性检测

通过测试程序，对初始状态的 PCI 密码卡进行密码运算功能检测，应检测失败；对就绪状态的 PCI 密码卡进行密码运算功能检测，应检测成功。

通过测试程序，对错误的私钥访问控制码，应检测失败；对正确的私钥访问控制码，应检测成功。

通过测试程序，对错误的用户 IC 卡或者 USBKey，应检测失败；对正确的用户 IC 卡或者 USBKey，应检测成功。

附录 A
（规范性附录）
PCI 密码卡 API 函数集

A.1　系统架构说明

PCI 密码卡 API 函数集是以动态库的方式提供给用户使用，任何支持 Win32 DLL 或 Linux 动态库的编程软件均可使用该 API 函数集。

该 API 函数集系统架构由以下两部分组成（如图示）：

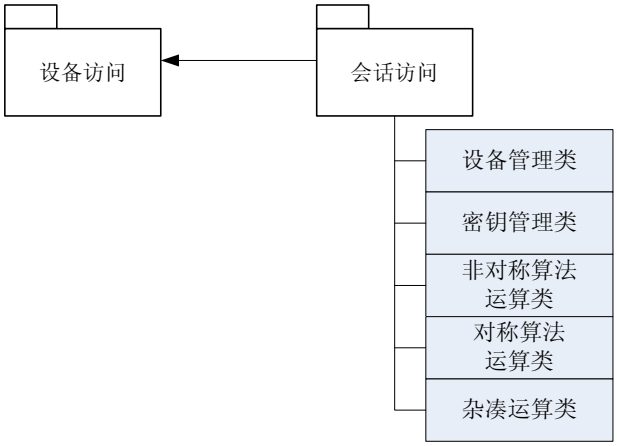

图 2　API 函数集系统架构图

A.1.1　设备访问

用户首先通过设备访问获得对 PCI 密码卡硬件设备的访问权限。

设备访问是整个 API 函数集的基础子系统。用户在使用接口时，首先必须使用该系统获得对硬件设备的访问权限，之后才能够继续后续的会话访问操作。

通常，一台主机只支持同一厂商同一型号的一个 PCI 密码卡硬件设备访问。

A.1.2　会话访问

会话访问是负责系统的安全功能访问操作的子系统。在获得设备访问后，用户可通过会话访问可实现设备基本管理、提取随机数、对称算法运算、非对称算法运算、数据块杂凑运算等安全功能的应用；而对于非对称密钥的私钥使用则可通过私钥访问控制码认证控制实现。

A.1.3　返回结果定义

整个 API 函数集接口函数调用时，如果返回值为 SDR_OK，表示该函数调用成功；否则调用失败。具体错误代码定义请参见附录 B。

A.2　算法和密钥定义

A.2.1　算法标识定义

对称算法标识		
宏描述	预定义值	说明
#define SGD_SM1_ECB	0x00000101	SM1 算法 ECB 加密模式
#define SGD_SM1_CBC	0x00000102	SM1 算法 CBC 加密模式
#define SGD_SM1_CFB	0x00000104	SM1 算法 CFB 加密模式
#define SGD_SM1_OFB	0x00000108	SM1 算法 OFB 加密模式
#define SGD_SM1_MAC	0x00000110	SM1 算法 MAC 加密模式
#define SGD_SSF33_ECB	0x00000201	SSF33 算法 ECB 加密模式
#define SGD_SSF33_CBC	0x00000202	SSF33 算法 CBC 加密模式

#define SGD_SSF33_CFB	0x00000204	SSF33 算法 CFB 加密模式
#define SGD_SSF33_OFB	0x00000208	SSF33 算法 OFB 加密模式
#define SGD_SSF33_MAC	0x00000210	SSF33 算法 MAC 加密模式
0x00000400—0x800000xx		为其它对称算法预留
非对称算法标识		
宏描述	预定义值	说明
#define SGD_RSA	0x00010000	RSA 算法
#define SGD_SM2_1	0x00020100	SM2 椭圆曲线签名算法
#define SGD_SM2_2	0x00020200	SM2 椭圆曲线密钥交换协议
#define SGD_SM2_3	0x00020400	SM2 椭圆曲线加密算法
0x00000400—0x800000xx		为其它非对称算法预留
杂凑算法标识		
宏描述	预定义值	说明
#define SGD_SM3	0x00000001	SM3 杂凑算法
#define SGD_SHA1	0x00000002	SHA1 杂凑算法
#define SGD_SHA256	0x00000004	SHA256 杂凑算法
0x00000040—0x00000080		为其它杂凑算法预留

A.2.2 设备信息定义

字段名称	数据长度（字节）	含义
IssuerName	40	设备生产厂商名称
DeviceName	16	设备型号
DeviceSerial	16	设备编号，包含：日期（8 字符）、批次号（3 字符）、流水号（5 字符）
DeviceVersion	4	密码设备内部软件的版本号
StandardVersion	4	密码设备支持的接口规范版本号
AsymAlgAbility	8	前 4 字节表示支持的算法,表示方法为非对称算法标识按位或的结果；后 4 字节表示算法的最大模长，表示方法为支持的模长按位或的结果
SymAlgAbility	4	所有支持的对称算法，表示方法为对称算法标识按位或运算结果
HashAlgAbility	4	所有支持的杂凑算法，表示方法为杂凑算法标识按位或运算结果
BufferSize	4	支持的最大文件存储空间（单位字节）

实际数据结构定义：

```
typedef struct DeviceInfo_st{
    unsigned char IssuerName[40];
    unsigned char DeviceName[16];
    unsigned char DeviceSerial[16]
    unsigned int  DeviceVersion;
    unsigned int  StandardVersion;
```

```
    unsigned int  AsymAlgAbility[2];
    unsigned int  SymAlgAbility;
    unsigned int  HashAlgAbility;
    unsigned int  BufferSize;
}DEVICEINFO;
```

A.2.3 密钥分类及存储定义

1) 设备密钥与用户密钥

密钥存储区，用于非对称密钥对的存放，索引号从 0 开始检索，每个索引号对应一个签名密钥对和一个加密密钥对。其中，索引号为 0 表示设备密钥对。索引号 1 开始表示用户密钥对。

设备密钥对只能在设备初始化时使用管理工具生成或安装，用户密钥对通过密码设备管理工具生成或安装。

密钥对索引号	公钥	私 钥
0x00	设备签名公钥	设备签名私钥
	设备加密公钥	设备加密私钥
0x01	用户签名公钥	用户签名私钥
	用户加密公钥	用户加密私钥
……	……	……
	……	……

2) 密钥加密密钥

密钥加密密钥通过密码设备管理工具生成或安装，存储区可存储密钥长度为 128 位的密钥加密密钥，使用索引号从 1 开始。

密钥索引号	密钥加密密钥
0x01	密钥加密密钥 001
……	……

3) 会话密钥

会话密钥使用本 API 函数集的设备接口函数生成或导入，会话密钥使用句柄检索。

A.3 基本结构定义

A.3.1 RSA 密钥数据结构定义

RSA 密钥结构存储时顺序为从高到低，即密钥存放时从密钥结构数组的最高位开始，最高字节填在最高位，不足位填充数据 0。

公钥数据结构定义		
字段名称	数据长度(字节)	含义
bits	4	模长
m	256	模 N
e	256	公开密钥 e

私钥数据结构定义

字段名称	数据长度(字节)	含义
bits	4	模长
m	256	模 N
e	256	公开密钥 e
d	256	私有密钥 d
prime[2]	128 * 2	素数 p 和 q
pexp[2]	128 * 2	Dp 和 Dq
coef	128	系数 i

实际数据结构定义：

```
#define RSAref_MAX_BITS    2048
#define RSAref_MAX_LEN     ((RSAref_MAX_BITS + 7) / 8)
#define RSAref_MAX_PBITS   ((RSAref_MAX_BITS + 1) / 2)
#define RSAref_MAX_PLEN    ((RSAref_MAX_PBITS + 7)/ 8)
typedef struct RSArefPublicKey_st
{
    unsigned int  bits;
    unsigned char m[RSAref_MAX_LEN];
    unsigned char e[RSAref_MAX_LEN];
} RSArefPublicKey;

typedef struct RSArefPrivateKey_st
{
    unsigned int  bits;
    unsigned char m[RSAref_MAX_LEN];
    unsigned char e[RSAref_MAX_LEN];
    unsigned char d[RSAref_MAX_LEN];
    unsigned char prime[2][RSAref_MAX_PLEN];
    unsigned char pexp[2][RSAref_MAX_PLEN];
    unsigned char coef[RSAref_MAX_PLEN];
} RSArefPrivateKey;
```

A.3.2 ECC 密钥数据结构定义

公钥数据结构定义		
字段名称	数据长度	含义
bits	4	模长
x	32	公钥 x 坐标
y	32	公钥 y 坐标

私钥数据结构定义		
字段名称	数据长度	含义
D	32	私钥

实际数据结构定义：

```
#define ECCref_MAX_BITS         256
#define ECCref_MAX_LEN          ((ECCref_MAX_BITS+7) / 8)
typedef struct ECCrefPublicKey_st
{
    unsigned int  bits;
    unsigned char x[ECCref_MAX_LEN];
    unsigned char y[ECCref_MAX_LEN];
} ECCrefPublicKey;

typedef struct ECCrefPrivateKey_st
{
    unsigned int  bits;
    unsigned char D[ECCref_MAX_LEN];
} ECCrefPrivateKey;
```

A.3.3 ECC 加密数据结构定义

加密数据结构定义		
字段名称	数据长度	含义
x	32	与 y 组成椭圆曲线上的点（x，y）
y	32	与 x 组成椭圆曲线上的点（x，y）
C	32	加密数据
M	32	预留，用于支持带 MAC 输出的 ECC 算法

实际数据结构定义：

```
typedef struct ECCCipher_st
{
    unsigned char x[ECCref_MAX_LEN];
    unsigned char y[ECCref_MAX_LEN];
    unsigned char C[ECCref_MAX_LEN];
    unsigned char M[ECCref_MAX_LEN];
} ECCCipher;
```

A.3.4 ECC 签名数据结构定义

签名数据结构定义		
字段名称	数据长度	含义
r	32	签名的 r 部分
s	32	签名的 s 部分

实际数据结构定义：

```
typedef struct ECCSignature_st
{
    unsigned char r[ECCref_MAX_LEN];
    unsigned char s[ECCref_MAX_LEN];
```

} ECCSignature;

A.4 API 函数集描述

API 函数集是 PCI 密码卡需要具备的功能集，其中部分功能函数可由各个生产厂商自行决定是否支持。根据功能的不同，分为以下功能模块：

1) 设备管理类函数
2) 密钥管理类函数
3) 非对称算法运算类函数
4) 对称算法运算类函数
5) 杂凑运算类函数
6) 文件管理类函数

A.4.1 设备管理类函数

设备管理类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
打开设备	SDF_OpenDevice
关闭设备	SDF_CloseDevice
创建会话	SDF_OpenSession
关闭会话	SDF_CloseSession
获取设备信息	SDF_GetDeviceInfo
产生随机数	SDF_GenerateRandom
获取私钥使用权限	SDF_GetPrivateKeyAccessRight
释放私钥使用权限	SDF_ReleasePrivateKeyAccessRight

函数 1： 打开设备

原型： int SDF_OpenDevice(void **phDeviceHandle);

描述： 打开密码设备。在使用 PCI 密码卡提供的功能之前，必须首先执行打开 PCI 密码卡操作。该操作完成 PCI 密码卡驱动程序的初始化并复位 PCI 密码卡。

参数： phDeviceHandle[out] 返回设备句柄

返回值： 0 成功

非 0 失败，返回错误代码

备注： phDeviceHandle 由函数初始化并填写内容

函数 2：关闭设备

原型： int SDF_CloseDevice(void *hDeviceHandle);

描述： 关闭密码设备，并释放相关资源。在最后一次访问 PCI 密码卡提供的功能之后，必须执行关闭 PCI 密码卡操作。该操作完成系统资源的释放工作。

参数： hDeviceHandle[in] 已打开的设备句柄

返回值： 0 成功

非 0 失败，返回错误代码

函数 3：创建会话

原型： int SDF_OpenSession(void *hDeviceHandle, void **phSessionHandle);

描述： 创建与密码设备的会话。

参数： hDeviceHandle[in] 已打开的设备句柄

phSessionHandle[out] 返回与密码设备建立的新会话句柄

返回值： 0 成功
非 0 失败，返回错误代码

函数 4：关闭会话

原型： int SDF_CloseSession(void *hSessionHandle);
描述： 关闭与密码设备已建立的会话，并释放相关资源。
参数： hSessionHandle [in] 与密码设备已建立的会话句柄
返回值： 0 成功
非 0 失败，返回错误代码

函数 5：获取设备信息

原型：
```
int SDF_GetDeviceInfo (
        void *hSessionHandle,
        DEVICEINFO *pstDeviceInfo);
```
描述： 获取密码设备标志信息，用户可以通过该操作辨识 PCI 密码卡，用来区分不同厂商生产的 PCI 密码卡、同一厂商生产的不同型号的 PCI 密码卡以及同一型号不同批次的 PCI 密码卡。
参数： hSessionHandle[in] 与设备建立的会话句柄
pstDeviceInfo [out] 设备能力描述信息，内容及格式见设备信息定义
返回值： 0 成功
非 0 失败，返回错误代码

函数 6：产生随机数

原型：
```
int SDF_GenerateRandom (
      void *hSessionHandle,
      unsigned int  uiLength,
      unsigned char *pucRandom);
```
描述： 获取指定长度的随机数
参数： hSessionHandle[in] 与设备建立的会话句柄
uiLength[in] 欲获取的随机数长度
pucRandom[out] 缓冲区指针，用于存放获取的随机数
返回值： 0 成功
非 0 失败，返回错误代码

函数 7：获取私钥使用权限

原型：
```
int SDF_GetPrivateKeyAccessRight (
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      unsigned char *pucPassword,
      unsigned int  uiPwdLength);
```
描述： 获取密码设备内部存储的指定索引私钥的使用权。
参数： hSessionHandle[in] 与设备建立的会话句柄
uiKeyIndex[in] 密码设备存储私钥的索引值
pucPassword[in] 使用的私钥访问控制码

	uiPwdLength[in]	私钥访问控制码长度，不少于 8 字节
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 本规范涉及密码设备存储的密钥对索引值的的起始索引值为 1，最大为 n，密码设备的实际存储容量决定 n 值。

函数 8：释放私钥使用权限

```
原型：  int SDF_ReleasePrivateKeyAccessRight (
            void *hSessionHandle,
            unsigned int  uiKeyIndex);
```

描述： 释放密码设备存储的指定索引私钥的使用授权

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyIndex[in]	密码设备存储私钥索引值
返回值：	0	成功
	非 0	失败，返回错误代码

A.4.2 密钥管理类函数

密钥管理类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
导出 RSA 签名公钥	SDF_ExportSignPublicKey_RSA
导出 RSA 加密公钥	SDF_ExportEncPublicKey_RSA
产生 RSA 非对称密钥对并输出	SDF_GenerateKeyPair_RSA
生成会话密钥并用内部 RSA 公钥加密输出	SDF_GenerateKeyWithIPK_RSA
生成会话密钥并用外部 RSA 公钥加密输出	SDF_GenerateKeyWithEPK_RSA
导入会话密钥并用内部 RSA 私钥解密	SDF_ImportKeyWithISK_RSA
基于 RSA 算法的数字信封转换	SDF_ExchangeDigitEnvelopeBaseOnRSA
导出 ECC 签名公钥	SDF_ExportSignPublicKey_ECC
导出 ECC 加密公钥	SDF_ExportEncPublicKey_ECC
产生 ECC 非对称密钥对并输出	SDF_GenerateKeyPair_ECC
生成会话密钥并用内部 ECC 公钥加密输出	SDF_GenerateKeyWithIPK_ECC
生成会话密钥并用外部 ECC 公钥加密输出	SDF_GenerateKeyWithEPK_ECC
导入会话密钥并用内部 ECC 私钥解密	SDF_ImportKeyWithISK_ECC
发起方生成密钥协商参数并输出	SDF_GenerateAgreementDataWithECC
发起方计算会话密钥	SDF_GenerateKeyWithECC
响应方产生协商参数并计算会话密钥	SDF_GenerateAgreementDataAndKeyWithECC
基于 ECC 算法的数字信封转换	SDF_ExchangeDigitEnvelopeBaseOnECC
生成会话密钥并用密钥加密密钥加密输出	SDF_GenerateKeyWithKEK
导入会话密钥并用密钥加密密钥解密	SDF_ImportKeyWithKEK
导入明文会话密钥	SDF_ImportKey
销毁会话密钥	SDF_DestoryKey

函数 1：导出 RSA 签名公钥

```
原型：  int SDF_ExportSignPublicKey_RSA(
            void *hSessionHandle,
```

```
        unsigned int  uiKeyIndex,
        RSArefPublicKey *pucPublicKey);
```

描述： 导出密码设备内部存储的指定索引位置的签名公钥

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyIndex[in]	密码设备存储的 RSA 密钥对索引值
	pucPublicKey[out]	RSA 公钥结构
返回值：	0	成功
	非 0	失败，返回错误代码

函数 2：导出 RSA 加密公钥

原型：
```
int SDF_ExportEncPublicKey_RSA(
        void *hSessionHandle,
        unsigned int  uiKeyIndex,
        RSArefPublicKey *pucPublicKey);
```

描述： 导出密码设备内部存储的指定索引位置的加密公钥

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyIndex[in]	密码设备存储的 RSA 密钥对索引值
	pucPublicKey[out]	RSA 公钥结构
返回值：	0	成功
	非 0	失败，返回错误代码

函数 3：产生 RSA 密钥对并输出

原型：
```
int SDF_GenerateKeyPair_RSA(
        void *hSessionHandle,
        unsigned int  uiKeyBits,
        RSArefPublicKey *pucPublicKey,
        RSArefPrivateKey *pucPrivateKey);
```

描述： 请求密码设备产生指定模长的 RSA 密钥对

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyBits [in]	指定密钥模长
	pucPublicKey[out]	RSA 公钥结构
	pucPrivateKey[out]	RSA 私钥结构
返回值：	0	成功
	非 0	失败，返回错误代码

函数 4：生成会话密钥并用内部 RSA 公钥加密输出

原型：
```
int SDF_GenerateKeyWithIPK_RSA (
        void *hSessionHandle,
        unsigned int uiIPKIndex,
        unsigned int uiKeyBits,
        unsigned char *pucKey,
        unsigned int *puiKeyLength,
        void **phKeyHandle);
```

描述： 生成会话密钥并用指定索引的内部加密公钥加密输出，同时返回密钥句柄

参数： hSessionHandle[in] 与设备建立的会话句柄
uiIPKIndex[in] 密码设备内部存储公钥的索引值
uiKeyBits[in] 指定产生的会话密钥长度
pucKey[out] 缓冲区指针，用于存放返回的密钥密文
puiKeyLength[out] 返回的密钥密文长度
phKeyHandle[out] 返回的密钥句柄
返回值： 0 成功
非 0 失败，返回错误代码
备注： 公钥加密数据时填充方式按照 PKCS#1 v1.5 的要求进行；返回的密钥不包含 IV 部分。

函数 5：生成会话密钥并用外部 RSA 公钥加密输出

原型：
```
int SDF_GenerateKeyWithEPK_RSA (
      void *hSessionHandle,
      unsigned int uiKeyBits,
      RSArefPublicKey *pucPublicKey,
      unsigned char *pucKey,
      unsigned int *puiKeyLength,
      void **phKeyHandle);
```
描述： 生成会话密钥并用外部公钥加密输出，同时返回密钥句柄
参数： hSessionHandle[in] 与设备建立的会话句柄
uiKeyBits[in] 指定产生的会话密钥长度
pucPublicKey[in] 输入的外部 RSA 公钥结构
pucKey[out] 缓冲区指针，用于存放返回的密钥密文
puiKeyLength[out] 返回的密钥密文长度
phKeyHandle[out] 返回的密钥句柄
返回值： 0 成功
非 0 失败，返回错误代码
备注： 公钥加密数据时填充方式按照 PKCS#1 v1.5 的要求进行；返回的密钥不包含 IV 部分。

函数 6：导入会话密钥并用内部 RSA 私钥解密

原型：
```
int SDF_ImportKeyWithISK_RSA (
      void *hSessionHandle,
      unsigned int uiISKIndex,
      unsigned char *pucKey,
      unsigned int *puiKeyLength,
      void **phKeyHandle);
```
描述： 导入会话密钥并用内部私钥解密，同时返回密钥句柄
参数： hSessionHandle[in] 与设备建立的会话句柄
uiISKIndex[in] 密码设备内部存储加密私钥的索引值，对应于加密时的公钥
pucKey[in] 缓冲区指针，用于存放输入的密钥密文
puiKeyLength[in] 输入的密钥密文长度

	phKeyHandle[out]	返回的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码
备注：	去填充方式与公钥加密时相对应。	

函数 7：基于 RSA 算法的数字信封转换

原型：
```
int SDF_ExchangeDigitEnvelopeBaseOnRSA(
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      RSArefPublicKey *pucPublicKey,
      unsigned char *pucDEInput,
      unsigned int  uiDELength,
      unsigned char *pucDEOutput,
      unsigned int  *puiDELength);
```

描述：将由内部加密公钥加密的会话密钥转换为由外部指定的公钥加密，可用于数字信封转换。

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyIndex[in]	密码设备存储的内部 RSA 密钥对索引值
	pucPublicKey [in]	外部 RSA 公钥结构
	pucDEInput [in]	缓冲区指针，用于存放输入的会话密钥密文
	uiDELength[in]	输入的会话密钥密文长度
	pucDEOutput[out]	缓冲区指针，用于存放输出的会话密钥密文
	puiDELength[out]	输出的会话密钥密文长度
返回值：	0	成功
	非 0	失败，返回错误代码

函数 8：导出 ECC 签名公钥

原型：
```
int SDF_ExportSignPublicKey_ECC(
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      ECCrefPublicKey *pucPublicKey);
```

描述：导出密码设备内部存储的指定索引位置的签名公钥

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyIndex[in]	密码设备存储的 ECC 密钥对索引值
	pucPublicKey[out]	ECC 公钥结构
返回值：	0	成功
	非 0	失败，返回错误代码

函数 9：导出 ECC 加密公钥

原型：
```
int SDF_ExportEncPublicKey_ECC(
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      ECCrefPublicKey *pucPublicKey);
```

描述：导出密码设备内部存储的指定索引位置的加密公钥

参数： hSessionHandle[in] 与设备建立的会话句柄
uiKeyIndex[in] 密码设备存储的 ECC 密钥对索引值
pucPublicKey[out] ECC 公钥结构
返回值： 0 成功
非 0 失败，返回错误代码

函数 10：产生 ECC 密钥对并输出

原型：
```
int SDF_GenerateKeyPair_ECC(
      void *hSessionHandle,
      unsigned int  uiAlgID,
      unsigned int  uiKeyBits,
      ECCrefPublicKey *pucPublicKey,
      ECCrefPrivateKey *pucPrivateKey);
```
描述： 请求密码设备产生指定类型和模长的 ECC 密钥对

参数： hSessionHandle[in] 与设备建立的会话句柄
uiAlgID[in] 指定算法标识
uiKeyBits [in] 指定密钥长度
pucPublicKey[out] ECC 公钥结构
pucPrivateKey[out] ECC 私钥结构

返回值： 0 成功
非 0 失败，返回错误代码

函数 11：生成会话密钥并用内部 ECC 公钥加密输出

原型：
```
int SDF_GenerateKeyWithIPK_ECC (
      void *hSessionHandle,
      unsigned int uiIPKIndex,
      unsigned int uiKeyBits,
      ECCCipher *pucKey,
      void **phKeyHandle);
```
描述： 生成会话密钥并用指定索引的内部 ECC 加密公钥加密输出，同时返回密钥句柄

参数： hSessionHandle[in] 与设备建立的会话句柄
uiIPKIndex[in] 密码设备内部存储公钥的索引值
uiKeyBits[in] 指定产生的会话密钥长度
pucKey[out] 缓冲区指针，用于存放返回的密钥密文
phKeyHandle[out] 返回的密钥句柄

返回值： 0 成功
非 0 失败，返回错误代码

备注： 返回的密钥不包含 IV 部分。

函数 12：生成会话密钥并用外部 ECC 公钥加密输出

原型：
```
int SDF_GenerateKeyWithEPK_ECC (
      void *hSessionHandle,
      unsigned int uiKeyBits,
      unsigned int  uiAlgID,
```

```
        ECCrefPublicKey *pucPublicKey,
        ECCCipher *pucKey,
        void **phKeyHandle);
```

描述： 生成会话密钥并用外部 ECC 公钥加密输出，同时返回密钥句柄

参数：		
	hSessionHandle[in]	与设备建立的会话句柄
	uiKeyBits[in]	指定产生的会话密钥长度
	uiAlgID[in]	外部 ECC 公钥的算法标识
	pucPublicKey[in]	输入的外部 ECC 公钥结构
	pucKey[out]	缓冲区指针，用于存放返回的密钥密文
	phKeyHandle[out]	返回的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 返回的密钥不包含 IV 部分。

函数 13：导入会话密钥并用内部 ECC 私钥解密

原型：
```
int SDF_ImportKeyWithISK_ECC (
      void *hSessionHandle,
      unsigned int uiISKIndex,
      ECCCipher *pucKey,
      void **phKeyHandle);
```

描述： 导入会话密钥并用内部 ECC 加密私钥解密，同时返回密钥句柄

参数：		
	hSessionHandle[in]	与设备建立的会话句柄
	uiISKIndex[in]	密码设备内部存储加密私钥的索引值，对应于加密时的公钥
	pucKey[in]	缓冲区指针，用于存放输入的密钥密文
	phKeyHandle[out]	返回的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码

函数 14：发起方生成密钥协商参数并输出

原型：
```
int SDF_GenerateAgreementDataWithECC (
      void *hSessionHandle,
      unsigned int uiISKIndex,
      unsigned int uiKeyBits,
      unsigned char *pucSponsorID,
      unsigned int uiSponsorIDLength,
      ECCrefPublicKey  *pucSponsorPublicKey,
      ECCrefPublicKey  *pucSponsorTmpPublicKey,
      void **phAgreementHandle);
```

描述： 使用 ECC 密钥协商算法，为计算会话密钥而产生协商参数，同时返回指定索引位置的 ECC 公钥、临时 ECC 密钥对的公钥及协商句柄。

参数：		
	hSessionHandle[in]	与设备建立的会话句柄
	uiISKIndex[in]	密码设备内部存储加密私钥的索引值，该私钥用于参与密钥协商

	uiKeyBits[in]	要求协商的密钥长度
	pucSponsorID[in]	参与密钥协商的发起方 ID 值
	uiSponsorIDLength[in]	发起方 ID 长度
	pucSelfPublicKey[out]	返回的发起方 ECC 公钥结构
	pucSelfTmpPublicKey[out]	返回的发起方临时 ECC 公钥结构
	phAgreementHandle[out]	返回的协商句柄，用于计算协商密钥
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 为协商会话密钥，协商的发起方应首先调用本函数。

如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量。

函数 15：发起方计算会话密钥

原型：
```
int SDF_GenerateKeyWithECC (
        void *hSessionHandle,
        unsigned char *pucResponseID,
        unsigned int uiResponseIDLength,
        ECCrefPublicKey *pucResponsePublicKey,
        ECCrefPublicKey *pucResponseTmpPublicKey,
        void *hAgreementHandle,
        void **phKeyHandle);
```

描述： 使用 ECC 密钥协商算法，使用自身协商句柄和响应方的协商参数计算会话密钥，同时返回会话密钥句柄。

参数：	hSessionHandle[in]	与设备建立的会话句柄
	pucResponseID[in]	外部输入的响应方 ID 值
	uiResponseIDLength[in]	外部输入的响应方 ID 长度
	pucResponsePublicKey[in]	外部输入的响应方 ECC 公钥结构
	pucResponseTmpPublicKey[in]	外部输入的响应方临时 ECC 公钥结构
	hAgreementHandle[in]	协商句柄，用于计算协商密钥
	phKeyHandle[out]	返回的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 协商的发起方获得响应方的协商参数后调用本函数，计算会话密钥。

如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量。

函数 16：响应方产生协商数据并计算会话密钥

原型：
```
int SDF_GenerateAgreementDataAndKeyWithECC (
        void *hSessionHandle,
        unsigned int uiISKIndex,
        unsigned int uiKeyBits,
        unsigned char *pucResponseID,
        unsigned int uiResponseIDLength,
        unsigned char *pucSponsorID,
        unsigned int uiSponsorIDLength,
        ECCrefPublicKey *pucSponsorPublicKey,
```

```
        ECCrefPublicKey *pucSponsorTmpPublicKey,
        ECCrefPublicKey  *pucResponsePublicKey,
        ECCrefPublicKey  *pucResponseTmpPublicKey,
        void **phKeyHandle);
```

描述： 使用 ECC 密钥协商算法，产生协商参数并计算会话密钥，同时返回产生的协商参数和和密钥句柄。

参数：

hSessionHandle[in]	与设备建立的会话句柄
uiISKIndex[in]	密码设备内部存储加密私钥的索引值，该私钥用于参与密钥协商
uiKeyBits[in]	协商后要求输出的密钥长度
pucResponseID[in]	响应方 ID 值
uiResponseIDLength[in]	响应方 ID 长度
pucSponsorID[in]	发起方 ID 值
uiSponsorIDLength[in]	发起方 ID 长度
pucSponsorPublicKey[in]	外部输入的发起方 ECC 公钥结构
pucSponsorTmpPublicKey[in]	外部输入的发起方临时 ECC 公钥结构
pucResponsePublicKey[out]	返回的响应方 ECC 公钥结构
pucResponseTmpPublicKey[out]	返回的响应方临时 ECC 公钥结构
phKeyHandle[out]	返回的密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注： 本函数由响应方调用。

如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量

函数 17：基于 ECC 算法的数字信封转换

原型：

```
int SDF_ExchangeDigitEnvelopeBaseOnECC(
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      unsigned int  uiAlgID,
      ECCrefPublicKey *pucPublicKey,
      ECCCipher *pucEncDataIn,
      ECCCipher *pucEncDataOut);
```

描述： 将由内部加密公钥加密的会话密钥转换为由外部指定的公钥加密，可用于数字信封转换。

参数：

hSessionHandle[in]	与设备建立的会话句柄
uiKeyIndex[in]	密码设备存储的 ECC 密钥对索引值
uiAlgID[in]	外部 ECC 公钥的算法标识
pucPublicKey [in]	外部 ECC 公钥结构
pucEncDataIn[in]	缓冲区指针，用于存放输入的会话密钥密文
pucEncDataOut[out]	缓冲区指针，用于存放输出的会话密钥密文

返回值：

0	成功
非 0	失败，返回错误代码

函数 18：生成会话密钥并用密钥加密密钥加密输出

原型：
```
int SDF_GenerateKeyWithKEK (
        void *hSessionHandle,
        unsigned int uiKeyBits,
        unsigned int  uiAlgID,
        unsigned int uiKEKIndex,
        unsigned char *pucKey,
        unsigned int *puiKeyLength,
        void **phKeyHandle);
```

描述：生成会话密钥并用密钥加密密钥加密输出，同时返回密钥句柄。

参数：

hSessionHandle[in]	与设备建立的会话句柄
uiKeyBits[in]	指定产生的会话密钥长度
uiAlgID[in]	算法标识，指定对称加密算法
uiKEKIndex[in]	密码设备内部存储密钥加密密钥的索引值
pucKey[out]	缓冲区指针，用于存放返回的密钥密文
puiKeyLength[out]	返回的密钥密文长度
phKeyHandle[out]	返回的密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注：加密模式使用 ECB 模式。

函数 19：导入会话密钥并用密钥加密密钥解密

原型：
```
int SDF_ImportKeyWithKEK (
        void *hSessionHandle,
        unsigned int  uiAlgID,
        unsigned int uiKEKIndex,
        unsigned char *pucKey,
        unsigned int *puiKeyLength,
        void **phKeyHandle);
```

描述：导入会话密钥并用密钥加密密钥解密，同时返回会话密钥句柄。

参数：

hSessionHandle[in]	与设备建立的会话句柄
uiAlgID[in]	算法标识，指定对称加密算法
uiKEKIndex[in]	密码设备内部存储密钥加密密钥的索引值
pucKey[in]	缓冲区指针，用于存放输入的密钥密文
puiKeyLength[in]	输入的密钥密文长度
phKeyHandle[out]	返回的密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注：加密模式使用 ECB 模式。

函数 20：导入明文会话密钥

原型：
```
int SDF_ImportKey (
        void *hSessionHandle,
        unsigned char *pucKey,
```

```
        unsigned int uiKeyLength,
        void **phKeyHandle);
```

描述： 导入明文会话密钥，同时返回密钥句柄

参数：	hSessionHandle[in]	与设备建立的会话句柄
	pucKey[in]	缓冲区指针，用于存放输入的密钥明文
	puiKeyLength[in]	输入的密钥明文长度
	phKeyHandle[out]	返回的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码

函数 21：销毁会话密钥

```
原型： int SDF_DestoryKey (
        void *hSessionHandle,
        void *hKeyHandle);
```

描述： 销毁会话密钥，并释放为密钥句柄分配的内存等资源。

参数：	hSessionHandle[in]	与设备建立的会话句柄
	hKeyHandle[in]	输入的密钥句柄
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 在对称算法运算完成后，应调用本函数销毁会话密钥。

A.4.3 非对称算法运算类函数

非对称算法运算类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
外部公钥 RSA 运算	SDF_ExternalPublicKeyOperation_RSA
外部私钥 RSA 运算	SDF_ExternalPrivateKeyOperation_RSA
内部签名公钥 RSA 运算	SDF_InternalPublicKeyOperation_RSA
内部签名私钥 RSA 运算	SDF_InternalPrivateKeyOperation_RSA
外部密钥 ECC 签名	SDF_ExternalSign_ECC
外部密钥 ECC 验证	SDF_ExternalVerify_ECC
内部签名密钥 ECC 签名	SDF_InternalSign_ECC
内部签名密钥 ECC 验证	SDF_InternalVerify_ECC
外部密钥 ECC 加密	SDF_ExternalEncrypt_ECC
外部密钥 ECC 解密	SDF_ExternalDecrypt_ECC

函数 1：外部公钥 RSA 运算

```
原型： int SDF_ExternalPublicKeyOperation_RSA(
        void *hSessionHandle,
        RSArefPublicKey *pucPublicKey,
        unsigned char *pucDataInput,
        unsigned int  uiInputLength,
        unsigned char *pucDataOutput,
        unsigned int  *puiOutputLength);
```

描述： 指定使用外部公钥对数据进行运算

参数： hSessionHandle[in] 与设备建立的会话句柄
pucPublicKey [in] 外部 RSA 公钥结构
pucDataInput [in] 缓冲区指针，用于存放输入的数据
uiInputLength[in] 输入的数据长度
pucDataOutput[out] 缓冲区指针，用于存放输出的数据
puiOutputLength[out] 输出的数据长度
返回值： 0 成功
非 0 失败，返回错误代码
备注： 数据格式由应用层封装

函数 2：外部私钥 RSA 运算

原型：
```
int SDF_ExternalPrivateKeyOperation_RSA(
      void *hSessionHandle,
      RSArefPrivateKey *pucPrivateKey,
      unsigned char *pucDataInput,
      unsigned int  uiInputLength,
      unsigned char *pucDataOutput,
      unsigned int  *puiOutputLength);
```
描述： 指定使用外部私钥对数据进行运算
参数： hSessionHandle[in] 与设备建立的会话句柄
pucPrivateKey [in] 外部 RSA 私钥结构
pucDataInput [in] 缓冲区指针，用于存放输入的数据
uiInputLength [in] 输入的数据长度
pucDataOutput [out] 缓冲区指针，用于存放输出的数据
puiOutputLength [out] 输出的数据长度
返回值： 0 成功
非 0 失败，返回错误代码
备注： 数据格式由应用层封装

函数 3：内部公钥 RSA 运算

原型：
```
int SDF_InternalPublicKeyOperation_RSA(
      void *hSessionHandle,
      unsigned int  uiKeyIndex,
      unsigned char *pucDataInput,
      unsigned int  uiInputLength,
      unsigned char *pucDataOutput,
      unsigned int  *puiOutputLength);
```
描述： 使用内部指定索引的公钥对数据进行运算
参数： hSessionHandle[in] 与设备建立的会话句柄
uiKeyIndex[in] 密码设备内部存储公钥的索引值
pucDataInput[in] 缓冲区指针，用于存放外部输入的数据
uiInputLength[in] 输入的数据长度
pucDataOutput[out] 缓冲区指针，用于存放输出的数据
puiOutputLength[out] 输出的数据长度

返回值： 0 成功
非 0 失败，返回错误代码
备注： 索引范围仅限于内部签名密钥对，数据格式由应用层封装

函数 4：内部私钥 RSA 运算

原型： int SDF_InternalPrivateKeyOperation_RSA(
void *hSessionHandle,
unsigned int uiKeyIndex,
unsigned char *pucDataInput,
unsigned int uiInputLength,
unsigned char *pucDataOutput,
unsigned int *puiOutputLength);
描述： 使用内部指定索引的私钥对数据进行运算
参数： hSessionHandle[in] 与设备建立的会话句柄
uiKeyIndex[in] 密码设备内部存储私钥的索引值
pucDataInput[in] 缓冲区指针，用于存放外部输入的数据
uiInputLength[in] 输入的数据长度
pucDataOutput[out] 缓冲区指针，用于存放输出的数据
puiOutputLength[out] 输出的数据长度
返回值： 0 成功
非 0 失败，返回错误代码
备注： 索引范围仅限于内部签名密钥对，数据格式由应用层封装

函数 5：外部密钥 ECC 签名

原型： int SDF_ExternalSign_ECC(
void *hSessionHandle,
unsigned int uiAlgID,
ECCrefPrivateKey *pucPrivateKey,
unsigned char *pucData,
unsigned int uiDataLength,
ECCSignature *pucSignature);
描述： 使用外部 ECC 私钥对数据进行签名运算
参数： hSessionHandle[in] 与设备建立的会话句柄
uiAlgID[in] 算法标识，指定使用的 ECC 算法
pucPrivateKey[in] 外部 ECC 私钥结构
pucData[in] 缓冲区指针，用于存放外部输入的数据
uiDataLength[in] 输入的数据长度
pucSignature[out] 缓冲区指针，用于存放输出的签名值数据
返回值： 0 成功
非 0 失败，返回错误代码
备注： 对原文的杂凑运算，在函数外部完成。

函数 6：外部密钥 ECC 验证

原型： int SDF_ExternalVerify_ECC(

```
        void *hSessionHandle,
        unsigned int uiAlgID,
        ECCrefPublicKey *pucPublicKey,
        unsigned char *pucDataInput,
        unsigned int  uiInputLength,
        ECCSignature *pucSignature);
```

描述： 使用外部 ECC 公钥对 ECC 签名值进行验证运算

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiAlgID[in]	算法标识，指定使用的 ECC 算法
	pucPublicKey[in]	外部 ECC 公钥结构
	pucData[in]	缓冲区指针，用于存放外部输入的数据
	uiDataLength[in]	输入的数据长度
	pucSignature[in]	缓冲区指针，用于存放输入的签名值数据
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 对原文的杂凑运算，在函数外部完成。

函数 7：内部密钥 ECC 签名

原型：
```
int SDF_InternalSign_ECC(
        void *hSessionHandle,
        unsigned int  uiISKIndex,
        unsigned char *pucData,
        unsigned int  uiDataLength,
        ECCSignature *pucSignature);
```

描述： 使用内部 ECC 私钥对数据进行签名运算

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiISKIndex [in]	密码设备内部存储的 ECC 签名私钥的索引值
	pucData[in]	缓冲区指针，用于存放外部输入的数据
	uiDataLength[in]	输入的数据长度
	pucSignature [out]	缓冲区指针，用于存放输出的签名值数据
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 对原文的杂凑运算，在函数外部完成。

函数 8：内部密钥 ECC 验证

原型：
```
int SDF_InternalVerify_ECC(
        void *hSessionHandle,
        unsigned int  uiISKIndex,
        unsigned char *pucData,
        unsigned int  uiDataLength,
        ECCSignature *pucSignature);
```

描述： 使用内部 ECC 公钥对 ECC 签名值进行验证运算

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiISKIndex [in]	密码设备内部存储的 ECC 签名公钥的索引值

	pucData[in]	缓冲区指针，用于存放外部输入的数据
	uiDataLength[in]	输入的数据长度
	pucSignature[in]	缓冲区指针，用于存放输入的签名值数据
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 对原文的杂凑运算，在函数外部完成。

函数 9：外部密钥 ECC 公钥加密

原型：
```
int SDF_ExternalEncrytp_ECC(
      void *hSessionHandle,
      unsigned int uiAlgID,
      ECCrefPublicKey *pucPublicKey,
      unsigned char *pucData,
      unsigned int  uiDataLength,
      ECCCipher *pucEncData);
```

描述： 使用外部 ECC 公钥对数据进行加密运算

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiAlgID[in]	算法标识，指定使用的 ECC 算法
	pucPublicKey[in]	外部 ECC 公钥结构
	pucData[in]	缓冲区指针，用于存放外部输入的数据
	uiDataLength[in]	输入的数据长度
	pucEncData[out]	缓冲区指针，用于存放输出的数据密文
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 输入的数据长度 uiDataLength 不大于 ECCref_MAX_LEN。

函数 10：外部密钥 ECC 私钥解密

原型：
```
int SDF_ExternalDecrypt_ECC(
      void *hSessionHandle,
      unsigned int uiAlgID,
      ECCrefPrivateKey *pucPrivateKey,
      ECCCipher *pucEncData,
      unsigned char *pucData,
      unsigned int  *puiDataLength);
```

描述： 使用外部 ECC 私钥进行解密运算

参数：	hSessionHandle[in]	与设备建立的会话句柄
	uiAlgID[in]	算法标识，指定使用的 ECC 算法
	pucPrivateKey[in]	外部 ECC 私钥结构
	pucEncData[in]	缓冲区指针，用于存放输入的数据密文
	pucData[out]	缓冲区指针，用于存放输出的数据明文
	puiDataLength[out]	输出的数据明文长度
返回值：	0	成功
	非 0	失败，返回错误代码

A.4.4 对称算法运算类函数

对称算法运算类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
对称加密运算	SDF_Encrypt
对称解密运算	SDF_Decrypt
消息认证码运算	SDF_MAC

函数 1：对称加密

```
原型：  int SDF_Encrypt(
            void *hSessionHandle,
            void *hKeyHandle,
            unsigned int uiAlgID,
            unsigned char *pucIV,
            unsigned char *pucData,
            unsigned int uiDataLength,
            unsigned char *pucEncData,
            unsigned int  *puiEncDataLength);
```

描述：　使用指定的密钥句柄和 IV 对数据进行对称加密运算

参数：

hSessionHandle[in]	与设备建立的会话句柄
hKeyHandle[in]	指定的密钥句柄
uiAlgID[in]	算法标识，指定对称加密算法
pucIV[in|out]	缓冲区指针，用于存放输入和返回的 IV 数据
pucData[in]	缓冲区指针，用于存放输入的数据明文
uiDataLength[in]	输入的数据明文长度
pucEncData[out]	缓冲区指针，用于存放输出的数据密文
puiEncDataLength[out]	输出的数据密文长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：　此函数不对数据进行填充处理，输入的数据必须是指定算法分组长度的整数倍。

函数 2：对称解密

```
原型：  int SDF_Decrypt (
            void *hSessionHandle,
            void *hKeyHandle,
            unsigned int uiAlgID,
            unsigned char *pucIV,
            unsigned char *pucEncData,
            unsigned int  uiEncDataLength,
            unsigned char *pucData,
            unsigned int *puiDataLength);
```

描述：　使用指定的密钥句柄和 IV 对数据进行对称解密运算

参数：

hSessionHandle[in]	与设备建立的会话句柄
hKeyHandle[in]	指定的密钥句柄
uiAlgID[in]	算法标识，指定对称加密算法

	pucIV[in|out]	缓冲区指针，用于存放输入和返回的 IV 数据
	pucEncData[in]	缓冲区指针，用于存放输入的数据密文
	uiEncDataLength[in]	输入的数据密文长度
	pucData[out]	缓冲区指针，用于存放输出的数据明文
	puiDataLength[out]	输出的数据明文长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 此函数不对数据进行填充处理，输入的数据必须是指定算法分组长度的整数倍。

函数 3：计算 MAC

原型：
```
int SDF_CalculateMAC(
      void *hSessionHandle,
      void *hKeyHandle,
      unsigned int uiAlgID,
      unsigned char *pucIV,
      unsigned char *pucData,
      unsigned int uiDataLength,
      unsigned char *pucMAC,
      unsigned int  *puiMACLength);
```

描述：	使用指定的密钥句柄和 IV 对数据进行 MAC 运算	
参数：	hSessionHandle[in]	与设备建立的会话句柄
	hKeyHandle[in]	指定的密钥句柄
	uiAlgID[in]	算法标识，指定 MAC 加密算法
	pucIV[in|out]	缓冲区指针，用于存放输入和返回的 IV 数据
	pucData[in]	缓冲区指针，用于存放输出的数据明文
	uiDataLength[in]	输出的数据明文长度
	pucMAC[out]	缓冲区指针，用于存放输出的 MAC 值
	puiMACLength[out]	输出的 MAC 值长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 此函数不对数据进行分包处理，多包数据 MAC 运算由 IV 控制最后的 MAC 值。

A.4.5 杂凑运算类函数

杂凑运算类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
杂凑运算初始化	SDF_HashInit
多包杂凑运算	SDF_HashUpdate
杂凑运算结束	SDF_HashFinal

函数 1：杂凑运算初始化

原型：
```
int SDF_HashInit(
      void *hSessionHandle,
      unsigned int uiAlgID
      ECCrefPublicKey *pucPublicKey,
```

unsigned char *pucID,
unsigned int uiIDLength);

描述： 三步式杂凑运算第一步。

参数：

hSessionHandle[in]	与设备建立的会话句柄
uiAlgID[in]	指定杂凑算法标识
pucPublicKey[in]	签名者的 ECC 公钥，产生用于 ECC 签名的杂凑值时有效
pucID[in]	签名者的 ID 值，产生用于 ECC 签名的杂凑值时有效
uiIDLength[in]	签名者的 ID 长度

返回值：

0	成功
非 0	失败，返回错误代码

备注： 如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量。

函数 2：多包杂凑运算

原型：
```
int SDF_HashUpdate(
      void *hSessionHandle,
      unsigned char *pucData,
      unsigned int  uiDataLength);
```

描述： 三步式杂凑运算第二步，对输入的明文进行杂凑运算

参数：

hSessionHandle[in]	与设备建立的会话句柄
pucData[in]	缓冲区指针，用于存放输入的数据明文
uiDataLength[in]	输入的数据明文长度

返回值：

0	成功
非 0	失败，返回错误代码

函数 3：杂凑运算结束

原型：
```
int SDF_HashFinal(
      void *hSessionHandle,
      unsigned char *pucHash,
      unsigned int  *puiHashLength);
```

描述： 三步式杂凑运算第三步，杂凑运算结束返回杂凑数据并清除中间数据

参数：

hSessionHandle[in]	与设备建立的会话句柄
pucHash[out]	缓冲区指针，用于存放输出的杂凑数据
puiHashLength[out]	返回的杂凑数据长度

返回值：

0	成功
非 0	失败，返回错误代码

A.4.6 用户文件操作类函数

用户文件操作类函数包括以下具体函数，各函数返回值见附录 B 错误代码定义：

功能说明	函数名
创建文件	SDF_CreateFile
读取文件	SDF_ReadFile
写入文件	SDF_WriteFile
删除文件	SDF_DeleteFile

函数 1：创建文件

原型： int SDF_CreateFile(
void *hSessionHandle,
unsigned char *pucFileName,
unsigned int uiNameLen,
unsigned int uiFileSize);

描述： 在密码设备内部创建用于存储用户数据的文件

参数：		
	hSessionHandle[in]	与设备建立的会话句柄
	pucFileName[in]	缓冲区指针，用于存放输入的文件名，最大长度 128 字节
	uiNameLen[in]	文件名长度
	uiFileSize[in]	文件所占存储空间的长度
返回值：	0	成功
	非 0	失败，返回错误代码

函数 2：读取文件

原型： int SDF_ReadFile(
void *hSessionHandle,
unsigned char *pucFileName,
unsigned int uiNameLen,
unsigned int uiOffset,
unsigned int *puiFileLength,
unsigned char *pucBuffer);

描述： 读取在密码设备内部存储用户数据的文件的内容

参数：		
	hSessionHandle[in]	与设备建立的会话句柄
	pucFileName[in]	缓冲区指针，用于存放输入的文件名，最大长度 128 字节
	uiNameLen[in]	文件名长度
	uiOffset[in]	指定读取文件时的偏移值
	puiFileLength[in|out]	入参时指定读取文件内容的长度；出参时返回实际读取文件内容的长度
	pucBuffer[out]	缓冲区指针，用于存放读取的文件数据
返回值：	0	成功
	非 0	失败，返回错误代码

函数 3：写文件

原型： int SDF_WriteFile(
void *hSessionHandle,
unsigned char *pucFileName,
unsigned int uiNameLen,
unsigned int uiOffset,
unsigned int uiFileLength,
unsigned char *pucBuffer);

描述： 向密码设备内部存储用户数据的文件中写入内容

参数：	hSessionHandle[in]	与设备建立的会话句柄
	pucFileName[in]	缓冲区指针，用于存放输入的文件名，最大长度128 字节
	uiNameLen[in]	文件名长度
	uiOffset[in]	指定写入文件时的偏移值
	uiFileLength[in]	指定写入文件内容的长度
	pucBuffer[in]	缓冲区指针，用于存放输入的写文件数据
返回值：	0	成功
	非 0	失败，返回错误代码

函数 4：删除文件

原型：
```
int SDF_DeleteFile(
      void *hSessionHandle,
      unsigned char *pucFileName,
      unsigned int uiNameLen);
```

描述： 删除指定文件名的密码设备内部存储用户数据的文件

参数：	hSessionHandle[in]	与设备建立的会话句柄
	pucFileName[in]	缓冲区指针，用于存放输入的文件名，最大长度128 字节
	uiNameLen[in]	文件名长度
返回值：	0	成功
	非 0	失败，返回错误代码

附录 B

（规范性附录）

错误代码定义

错误代码标识		
宏描述	预定义值	说明
#define SDR_OK	0x0	操作成功
#define SDR_BASE	0x01000000	错误码基础值
#define SDR_UNKNOWERR	SDR_BASE + 0x00000001	未知错误
#define SDR_NOTSUPPORT	SDR_BASE + 0x00000002	不支持的接口调用
#define SDR_COMMFAIL	SDR_BASE + 0x00000003	与设备通信失败
#define SDR_HARDFAIL	SDR_BASE + 0x00000004	运算模块无响应
#define SDR_OPENDEVICE	SDR_BASE + 0x00000005	打开设备失败
#define SDR_OPENSESSION	SDR_BASE + 0x00000006	创建会话失败
#define SDR_PARDENY	SDR_BASE + 0x00000007	无私钥使用权限
#define SDR_KEYNOTEXIST	SDR_BASE + 0x00000008	不存在的密钥调用
#define SDR_ALGNOTSUPPORT	SDR_BASE + 0x00000009	不支持的算法调用
#define SDR_ALGMODNOTSUPPORT	SDR_BASE + 0x0000000A	不支持的算法模式调用
#define SDR_PKOPERR	SDR_BASE + 0x0000000B	公钥运算失败
#define SDR_SKOPERR	SDR_BASE + 0x0000000C	私钥运算失败
#define SDR_SIGNERR	SDR_BASE + 0x0000000D	签名运算失败
#define SDR_VERIFYERR	SDR_BASE + 0x0000000E	验证签名失败
#define SDR_SYMOPERR	SDR_BASE + 0x0000000F	对称算法运算失败
#define SDR_STEPERR	SDR_BASE + 0x00000010	多步运算步骤错误
#define SDR_FILESIZEERR	SDR_BASE + 0x00000011	文件长度超出限制
#define SDR_FILENOEXIST	SDR_BASE + 0x00000012	指定的文件不存在
#define SDR_FILEOFSERR	SDR_BASE + 0x00000013	文件起始位置错误
#define SDR_KEYTYPEERR	SDR_BASE + 0x00000014	密钥类型错误
#define SDR_KEYERR	SDR_BASE + 0x00000015	密钥错误
… …	SDR_BASE + 0x00000016 至 SDR_BASE + 0x00FFFFFF	预留

参考文献

[1] 《商用密码管理条例》

[2] GB/T 15277－1994 《信息处理 64bit 分组密码算法的工作方式》

[3] GB/T 9813－2000 《微型计算机通用规范》

[4] ISO/IEC 7816-3－1997《Electronic signals and transmission protocols》

[5] GB/T 17903.3-1999 信息技术 安全技术 密钥管理 第 1 部分：框架

[6] GB/T 17903.3-1999 信息技术 安全技术 密钥管理 第 2 部分：使用对称技术的机制

[7] GB/T 17903.3-1999 信息技术 安全技术 密钥管理 第 3 部分：使用非对称技术的机制

[8] GB/T 17964-2000 信息技术 安全技术 加密算法 第 1 部分：概述

[9] GB/T 17964-2000 信息技术 安全技术 加密算法 第 2 部分：非对称加密

[10] GB/T 17964-2000 信息技术 安全技术 加密算法 第 3 部分：对称加密

[11] GB/T 18238.1-2000 信息技术 安全技术 散列函数 第 1 部分:概述

[12] GB/T 18238.2-2002 信息技术 安全技术 散列函数 第 2 部分:采用 n 位块密码的散列函数

[13] GB/T 18238.3-2002 信息技术 安全技术 散列函数 第 3 部分:专用散列函数

[14] GB/T XXXXX-200X 信息技术 安全技术 密码术语

[15] GB/T XXXXX-200X 公钥密码基础设施应用技术体系 密码设备应用接口规范

[16] 《随机性检测规范》