ICS 35.040
L 80
备案号：

GM

中华人民共和国密码行业标准

GM/T 0017—2012

智能密码钥匙
密码应用接口数据格式规范

Smart token cryptography application interface data format specification

2012-11-22 发布　　2012-11-22 实施

国家密码管理局　　发布

中华人民共和国密码
行业标准
智能密码钥匙
密码应用接口数据格式规范
GM/T 0017—2012
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100013)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)64275323 发行中心:(010)51780235
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 0.00 字数 00 千字
2012年 月第一版 2012年 月第一次印刷
*
书号:155066·2- 定价 00.00 元

GM/T 0017-2012

目　　次

前　言

本标准依据 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准的附录 A、附录 B 为规范性附录，附录 C 为资料性附录。

本标准由国家密码管理局提出并归口。

本标准主要起草单位：北京江南天安科技有限公司、北京握奇智能科技有限公司、北京飞天诚信科技有限公司、北京天地融科技有限公司、恒宝股份有限公司、北京数字证书认证中心有限公司、北京天威诚信电子商务服务有限公司、北京国富安电子商务安全认证有限公司。

本标准参与制定单位：北京海泰方圆科技有限公司，北京华大智宝电子系统有限公司，北京大明五洲科技有限公司，中钞信用卡产业发展有限公司，北京华虹集成电路设计有限责任公司，北京旋极信息技术股份有限公司，北京创原天地科技有限公司，中铁信安（北京）信息安全技术有限公司，北京天诚盛业科技有限公司，东方口岸科技有限公司、北京格尔世纪智能卡科技有限公司，北京永新视博数字电视技术有限公司，吉大正元信息技术股份有限公司，深圳市文鼎创数据科技有限公司，武汉天喻信息产业股份有限公司。

本标准主要起草人：刘平、王艳平、李少雄、刘波、李庆、邓小四、汪雪林、李国、胡衍分、朱鹏飞、赵李明、冯承勇、张海松、付伟。

引　言

国家密码管理局发布的GM/T BBBB《智能密码钥匙密码应用接口规范》，在应用层为国内智能密码钥匙的使用提供了统一的技术标准和接口规范，取得了良好的效果。为了更好地解决此接口规范与不同设备提供商的产品兼容性问题，在设备访问层提供统一的接口数据格式，编制《智能密码钥匙密码应用接口数据格式规范》很有必要。本标准在GM/T BBBB《智能密码钥匙密码应用接口规范》的基础上进一步规定了这类产品的数据访问接口，从数据类型、数据格式、参数描述和定义、安全性要求等方面进行了具体描述，可用于指导相关产品的研制、使用和检测。

本标准涉及的密码算法按照国家密码管理部门的要求使用。

智能密码钥匙
密码应用接口数据格式规范

1 范围

本标准规定了基于PKI密码体系的智能密码钥匙应用接口数据格式，给出了接口相关数据的类型、格式、参数的定义和描述、安全性要求。

本标准适用于智能密码钥匙产品的研制、使用和检测。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GM/T 0005 随机性检测规范

GM/T 0006 密码应用标识规范

GM/T AAAA SM2密码算法使用规范

GM/T BBBB 智能密码钥匙密码应用接口规范

ISO 7816-4-识别卡 带触点的集成电路卡 第四部分：组织、安全和交换命令

PKCS #1-RSA 实验室，RSA加密标准，v2.1，2002.7

Specification for Integrated Circuit(s) Cards Interface Devices，Revision 1.1，2005

3 术语和定义

以下术语和定义适用于本标准。

3.1

智能密码钥匙 smart token

能完成密码功能和安全存储的终端密码产品，一般采用USB接口。

3.2

设备 device

本标准中将智能密码钥匙统称为设备。

3.3

命令 command

应用接口向设备发出的一条信息，该信息启动一个操作或请求一个应答。

3.4

响应 response

设备处理完成收到的命令报文后，返回给应用接口的报文。

3.5

功能 function

由一个或多个命令实现的处理过程，其操作结果用于完成全部或部分交易。

3.6

消息鉴别码　message authentication code;MAC

消息鉴别算法的输出。

3.7

管理员　PIN Administrator PIN

管理员的口令,为 ASCII 字符串。

3.8

用户　PIN User PIN

用户的个人口令,为 ASCII 字符串。

3.9

应用　application

包括容器、设备认证密钥和文件的一种结构,具备独立的权限管理。

3.10

容器　container

密码设备中用于保存密钥所划分的唯一性存储空间。

3.11

设备认证　device authentication

智能密码钥匙对应用程序的认证。

3.12

设备认证密钥　device authentication key

用于设备认证的密钥。

3.13

设备标签　device label

设备的别名,可以由用户进行设定并存储于设备内部。

3.14

SM1 算法　SM1 algorithm

一种分组加密算法,分组长度为 128 比特,密钥长度为 128 比特。

3.15

SM2 算法　SM2 algorithm

一种椭圆曲线密码算法,密钥长度为 256 比特。

3.16

SM3 算法　SM3 algorithm

一种杂凑算法,输出长度为 256 比特。

3.17

SM4 算法　SM4 algorithm

一种分组加密算法,分组长度为 128 比特,密钥长度为 128 比特。

4　缩略语

下列缩略语和相应解释适用于本标准:

CBC　　密码块链接(Cipher Block Chaining);

ECB　　电子密码本(Electronic Code Book);

MAC　　报文鉴别代码(Message Authentication code);

SHA-1	安全哈希算法(Secure Hash Algorithm),摘要输出长度为 20 字节;
SHA-256	安全哈希算法(Secure Hash Algorithm),摘要输出长度为 32 字节;
PIN	个人识别码(Personal Identification Number);
PKI	公钥基础设施(Public Key Infrastructure)
PKCS	公钥密码使用标准(the Public-Key Cryptograthy Standard)
PKCS#1	公钥密码使用标准系列规范中的第 1 部分,定义 RSA 公开密钥算法加密和签名机制(The Public-Key Cryptography Standard Part 1)
API	应用编程接口(Application Programming Interface)
COS	卡操作系统(Card Operating System)
USB	通用串行总线(Universal Serials BUS)
ICC	集成电路卡(Integrated Circuit Card)
HID	人机接口设备(Human Interface Device),常用于鼠标、键盘等外设。
CCID	集成电路卡接口设备(Integreted Circuits Card Interface Device),CCID 标准规定了 CCID 设备是一种芯片/智能卡接口设备,设备通过 USB 接口与主机或其它嵌入式主机连接,进行符合 CCID 标准的数据通讯,同时设备通过符合 7816 标准协议的接口与智能卡进行通讯。
UMS	大容量存储设备(USB Mass Storage),常用于 U 盘和移动硬盘等外设。
APDU	应用协议数据单元(Application Protocol Data Unit)
CLA	APDU 的类型字节(Class Type)
INS	APDU 的命令字节(Instruction Byte of Command Message)
P1	APDU 命令头中的参数 1(Parameter 1)
P2	APDU 命令头中的参数 2(Parameter 2)
Lc	APDU 命令的报文数据域长度
Le	APDU 命令的期望返回数据
SW1	APDU 命令的返回状态码 1(Status Word One)
SW2	APDU 命令的返回状态码 2(Status Word Two)
RFU	保留给未来使用(Reserved for future use)

5 记号

下列记号表示法适用于本部分规范:

“0”至“9”和“A”至“F”表示 16 个十六进制数字;

(B_1)	表示字节(B_1)的值
$B_1 \parallel B_2$	表示字节 B_1(最高有效字节)和 B_2(最低有效字节)的并置
$(B_1 \parallel B_2)$	表示字节 B_1 和 B_2 并置的值
0x00～0x0F	表示十六进制数
XX	表示 1 个字节 16 进制数
XXXX	表示 2 个字节 16 进制数
XX…XX	表示若干个字节 16 进制数
#	表示编号

6 结构模型

智能密码钥匙应用接口数据格式位于智能密码钥匙应用程序接口与设备之间,如图 1 所示。

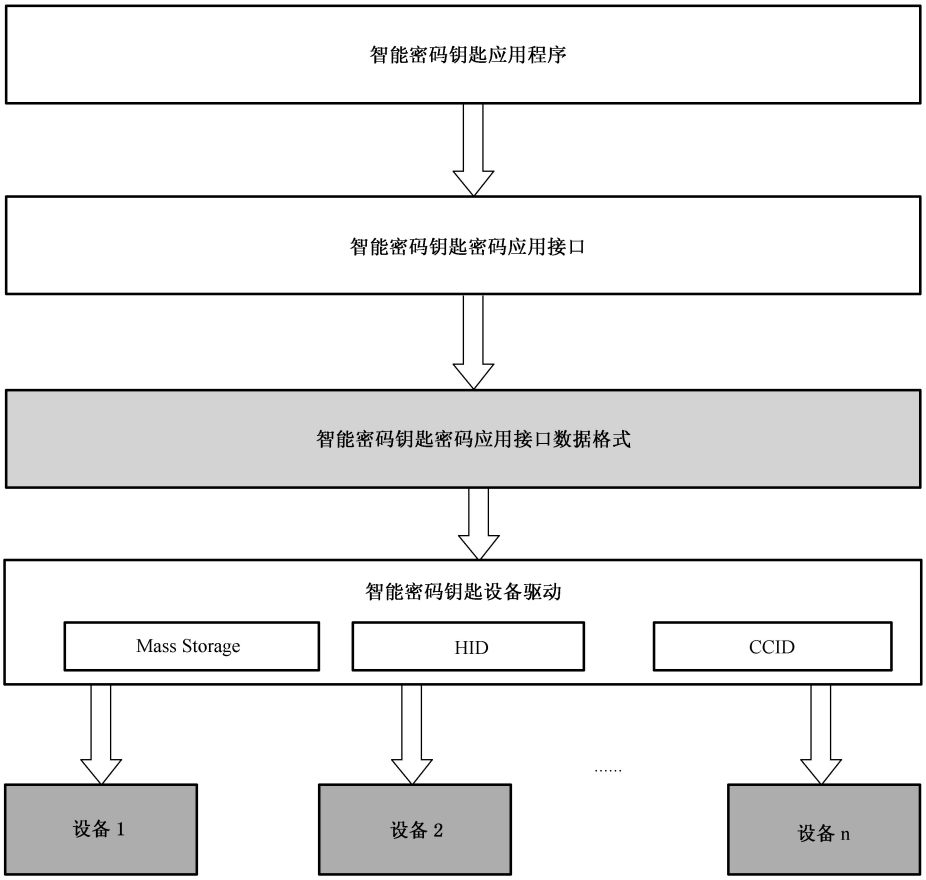


图 1　数据格式规范在应用层次关系中的位置

7　APDU 报文结构

7.1　概述

智能密码钥匙应用接口与设备之间的数据交换以 APDU 的形式进行编码。

应用协议中的一个步骤由发送命令、接收实体处理它以及发回响应组成。因此，特定的响应对应于特定的命令，称作为命令响应对。

APDU 可包含有命令报文或响应报文，它从接口设备发送到密码钥匙，或者相反地由密码钥匙发送到接口设备。

在命令响应对中，命令报文和响应报文都可以包含有数据，由表 1 所示的四种情况。

表 1　命令响应对内的数据

情　　况	命令数据	期望响应的数据
1	无数据	无数据
2	无数据	有数据
3	有数据	无数据
4	有数据	有数据

7.2 命令 APDU

如图 2 所示，本部分所定义的命令 APDU 由下列内容组成：

——必备的 4 字节命令头(CLA INS P1 P2)；

——有条件的可变长度主体。

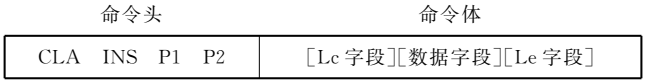


图 2 命令 APDU 结构

在命令 APDU 的数据字段中出现的字节数用 Lc 来表示。

在响应 APDU 的数据字段中期望的字节最大数用 Le(期望数据的长度)来表示。当 Le 字段只包含 0 时，则要求有效数据字节的最大数。

图 3 按照表 1 定义的 4 种情况示出了命令 APDU 的 4 种结构。

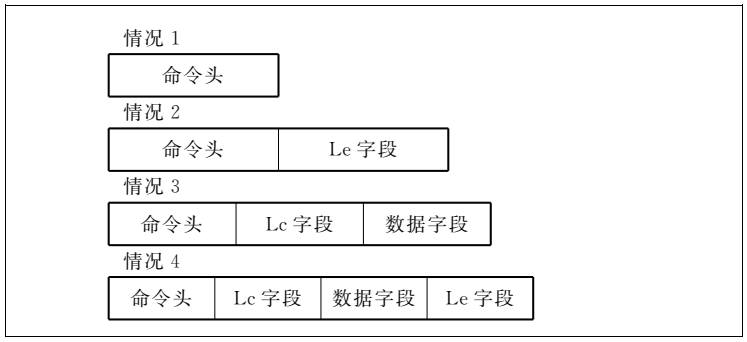


图 3 命令 APDU 的 4 种结构

在情况 1 时，长度 Lc 为 0，因此 Lc 字段和数据字段都为空。长度 Le 也为 0，因此 Le 字段为空。从而，命令体为空。

在情况 2 时，长度 Lc 为 0，因此 Lc 字段和数据字段都为空。长度 Le 不为 0，因此 Le 字段存在。从而，命令体由 Le 字段组成。

在情况 3 时，长度 Lc 不为 0，因此 Lc 字段存在，并且数据字段由 Lc 后续字节组成。长度 Le 为 0，因此 Le 字段为空。从而命令体由 Lc 字段后紧跟着数据字段组成。

在情况 4 时，长度 Lc 不为 0，因此 Lc 字段存在，并且数据字段由 Lc 后续字节组成。长度 Le 也不为 0，因此 Le 字段也存在。从而命令体由 Lc 字段后紧跟着数据字段和 Le 字段组成。

7.3 命令体的编码约定

在情况 1 时，命令 APDU 的命令体为空。这种命令 APDU 未带长度字段。

在情况 2、3 和 4 时，命令 APDU 的命令体由 B1 至 BL 所表示的 L 字节组成，如图 4 所示。这种命令体运载了 1 个或 2 个长度字段；B1 是第 1 个长度字段的一部分。

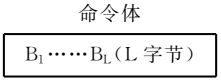


图 4 不为空的命令体

本标准要求 Lc 字段和 Le 字段必须为扩充方式编码(B1 的值为'00'，并且每个长度值都按 2 个其他字节进行编码)。

表 2 示出了按照表 1 和图 3 中定义的四种情况及可能的 Lc、Le 扩展的命令 APDU 的编码。

表 2　命令 APDU 的编码

条　件			情　况
L=0	—	—	1
L=3；	$(B_1)=0$；	—	2
L=3+(B_2 ‖ B_3)；	$(B_1)=0$；	(B_2 ‖ B_3)≠0	3
L=5+(B_2 ‖ B_3)；	$(B_1)=0$；	(B_2 ‖ B_3)≠0	4

任何其他命令 APDU 为无效的。

对以上四种情况说明如下。

情况 1——L=0；主体为空。

- 没有字节用于值为 0 的 Lc。
- 没有数据字节存在。
- 没有字节用于值为 0 的 Le。

情况 2——L=3，并且$(B_1)=0$

- 没有字节用于值为 0 的 Lc 。
- 没有数据字节存在。
- Le 字段由 3 个字节组成，其中 $B_1=0$，B_2 和 B_3 为 Le 的实际值，B_2 为高位字节。若 Le 为 0，则期望返回数据的 65 536 字节。

情况 3——L=3+(B_2 ‖ B_3)，$(B_1)=0$，并且(B_2 ‖ B_3)≠0。

- Lc 字段由前 3 个字节组成，其中，B_2 和 B_3 为 Lc 的实际值(1～65535)，B_2 为高位字节。
- B_4 至 BL 是数据字段中的 Lc 字节。
- 没有字节用于值为 0 的 Le。

情况 4——L=5+(B2 ‖ B3)，(B1)=0，并且(B2 ‖ B3)≠0。

- Lc 字段由前 3 个字节组成，其中 $B_1=0$，B_2 和 B_3 为 Lc(≠0)的实际值(1～65535)，B_2 为高位字节。
- B_4 至 BL-2 是数据字段中的 Lc 字节。
- Le 字段由最后的 2 个字节 BL-1 和 BL 组成，BL-1 为高位字节。若 Le 为 0，则期望返回数据的 65 536 字节。

7.4　响应 APDU

如图 5 所示，本规范定义的响应 APDU 由下列内容组成：

——有条件的可变长度主体；

——必备的 2 字节状态字(SW1-SW2)。

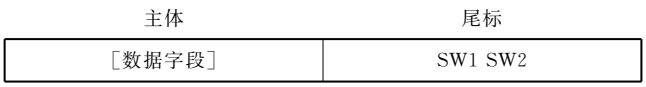


图 5　响应 APDU 结构

在响应 APDU 的数据字段中呈现的字节数用 Lr 来表示。

状态字对“命令响应对”之后的接收实体的状态进行了编码。

注：如果该命令被放弃，则响应 APDU 是一个状态字，它按 2 个状态字节来编码出错条件。

8 命令头、数据字段和响应状态字的编码约定

8.1 概述

表 3 示出了命令 APDU 的内容

表 3 命令 APDU 内容

代码	名称	长度(字节)	描　　述
CLA	类别	1	指令的类别
INS	指令	1	指令代码
P1	参数 1	1	指令参数 1
P2	参数 2	1	指令参数 2
Lc 字段	长度	3	在命令的数据字段中呈现的字节数
数据字段	数据	变量＝Lc	在命令的数据字段中发送的字节串
Le 字段	长度	变量＝2 或 3	在向命令响应的数据字段中期望的字节最大数

表 4 示出了响应 APDU 的内容

表 4 响应 APDU 内容

代码	名称	长度(字节)	描　　述
数据字段	数据	变量＝Lr	在响应的数据字段中收到的字节串
SW1	状态字节 1	1	命令处理状态
SW2	状态字节 2	1	命令处理受限字符

后续条规定了类别字节、指令字节、参数字节、数据字段字节和状态字节用的编码约定。

除非另有规定,在这些字节中,RFU 位都编码为 0,并且 RFU 字节也都编码为‘00’。

8.2 CLA(类别)字节

CLA 表示指令的类别。表 5 给出了本标准 CLA 的编码及含义。

表 5 CLA 的编码及含义

b8	b7	b6	b5	b4　b3	b2　b1	含　　义
1	0	0	x	—　—	—　—	命令链控制位
1	0	0	0	—　—	—　—	——表示单条指令或此命令为命令链的最后一条指令
1	0	0	1	—　—	—　—	——表示命令链中非最后的一条指令

命令链可用于支持需多步骤完成的任务或传输大数据量的场合。

8.3 INS(指令)字节

INS 表示需处理的命令。根据 7816-4 规范,‘6X’和‘9X’为非法的 INS 值。

表 6 示出了本标准定义的 INS 代码。

表 6　本标准定义的 INS 代码

值	命令名称	章　　节
‘02’	SetLabel	9.1.2
‘04’	GetDevInfo	9.1.3
‘10’	DevAuth	9.2.2
‘12’	ChangeDevAuthKey	9.2.3
‘14’	GetPinInfo	9.2.4
‘16’	ChangePin	9.2.5
‘18’	VerifyPin	9.2.6
‘1A’	UnblockPin	9.2.7
‘1C’	ClearSecureState	9.2.8
‘20’	CreateApplication	9.3.2
‘22’	EnumApplication	9.3.3
‘24’	DeleteApplication	9.3.4
‘26’	OpenApplication	9.3.5
‘28’	CloseApplication	9.3.6
‘30’	CreateFile	9.4.2
‘32’	DeleteFile	9.4.3
‘34’	EnumFiles	9.4.4
‘36’	GetFileInfo	9.4.5
‘38’	ReadFile	9.4.6
‘3A’	WriteFile	9.4.7
‘40’	CreateContainer	9.5.2
‘42’	OpenContainer	9.5.3
‘44’	CloseContainer	9.5.4
‘46’	EnumContainer	9.5.5
‘48’	DeleteContainer	9.5.6
‘4A’	GetContainerInfo	9.5.7
‘4C’	ImportCertificate	9.5.8
‘4E’	ExportCertificate	9.5.9
‘50’	GenRandom	9.6.2
‘52’	GenExtRSAKey	9.6.3
‘54’	GenRSAKeyPair	9.6.4
‘56’	ImportRSAKeyPair	9.6.5
‘58’	RSASignData	9.6.6
‘5E’	RSAVerify	9.6.7
‘5A’	RSAExportSessionKey	9.6.8
‘5C’	RSAExportSessionKeyEx	9.6.9

表 6（续）

值	命令名称	章　节
‘60’	ExtRSAPubKeyOperation	9.6.10
‘62’	ExtRSAPriKeyOperation	9.6.11
‘70’	GenECCKeyPair	9.6.12
‘72’	ImportECCKeyPair	9.6.13
‘74’	ECCSignData	9.6.14
‘76’	ECCVerify	9.6.15
‘78’	ECCExportSessionKey	9.6.16
‘80’	ECCExportSessionKeyEx	9.6.17
‘7A’	ExtECCEncrypt	9.6.18
‘7C’	ExtECCDecrypt	9.6.19
‘7E’	ExtECCSign	9.6.20
‘82’	GenerateAgreementDataWithECC	9.6.21
‘84’	GenerateAgreementDataAndKeyWithECC	9.6.22
‘86’	GenerateKeyWithECC	9.6.23
‘88’	ExportPubKey	9.6.24
‘A0’	ImportSessionKey	9.6.25
‘A2’	ImportSymmKey	9.6.26
‘A4’	EncryptInit	9.6.27
‘A6’	Encrypt	9.6.28
‘A8’	EncryptUpdate	9.6.29
‘AA’	EncryptFinal	9.6.30
‘AC’	DecryptInit	9.6.31
‘AE’	Decrypt	9.6.32
‘B0’	DecryptUpdate	9.6.33
‘B2’	DecryptFinal	9.6.34
‘B4’	DigestInit	9.6.35
‘B6’	Digest	9.6.36
‘B8’	DigestUpdate	9.6.37
‘BA’	DigestFinal	9.6.38
‘BC’	MacInit	9.6.39
‘BE’	Mac	9.6.40
‘C0’	MacUpdate	9.6.41
‘C2’	MacFinal	9.6.42
‘C4’	DestorySessionKey	9.6.43

8.4　参数字节

命令中的参数字节 P1-P2 可以具有任何值。如果参数字节不提供进一步的限定，则它应置为‘00’。

8.5 数据字段字节

在数据字段(即 DATA 域中的数据)中对 GM/T BBBB 或本标准中定义的数据结构(复合数据类型)进行传输时,结构体中的基本数据类型和大整数均以 BigEndian(高位字节在前)方式进行编码,整个结构体以紧缩格式进行编码(即各数据元素之间以单字节对齐)。

8.6 状态字节

响应的状态字节 SW1-SW2 表示了密码钥匙内的处理状态。图 6 示出了本标准定义的值的结构方案。

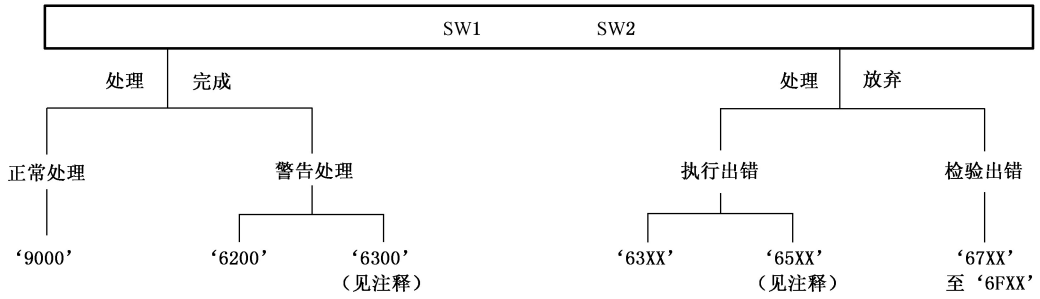


图 6 状态字节的结构方案

注:当 SW1=‘63’或‘65’时,非易失存储器的状态变化。当 SW1=除‘63’和‘65’外的‘6X’时,非易失存储器的状态不变化。

通过表 7 示出了本标准定义的 SW1-SW2 值的一般含义。对于每个命令,相应的条款提供了更详细的含义。

当 SW1 的值为‘65’,‘68’,‘69’和‘6A’时,表 8~表 11 规定了 SW2 的值。

表 7 SW1-SW2 的编码

SW1-SW2	含 义
	正常的处理
‘9000’	——处理正常,无进一步限定
	执行出错
‘64XX’	——非易失存储器状态不变化 (SW2=‘00’,其他值都是 RFU)
‘65XX’	——非易失存储器状态变化 (在 SW2 中进一步的限定,见表 8)
	校验出错
‘6700’	——错误的长度
‘68XX’	——CLA 的功能不被支持 (在 SW2 中进一步的限定,见表 9)
‘69XX’	——不允许的命令 (在 SW2 中进一步的限定,见表 10)
‘6AXX’	——错误的参数 P1~P2 (在 SW2 中进一步的限定,见表 11)
‘6B00’	——错误的参数 P1~P2
‘6CXX’	——错误的长度 Le;SW2 指示准确的长度
‘6D00’	——指令代码不被支持或无效
‘6E00’	——类别不被支持
‘6F00’	——没有精确的诊断

表 8　当 SW1＝‘65’时，SW2 的编码

SW2	含　义
‘00’	没有信息被给出
‘81’	写 EEPROM 出错

表 9　当 SW1＝‘68’时，SW2 的编码

SW2	含　义
‘00’	没有信息被给出
‘81’	逻辑信道不被支持
‘82’	安全报文不被支持

表 10　当 SW1＝‘69’时，SW2 的编码

SW2	含　义
‘00’	没有信息被给出
‘81’	命令与文件结构不兼容
‘82’	安全状态不被满足
‘83’	认证方法被锁定
‘84’	引用的数据无效
‘85’	使用的条件不被满足
‘86’	命令不被允许

表 11　当 SW1＝‘6A’时，SW2 的编码

SW2	含　义
‘00’	没有信息被给出
‘80’	在数据字段中的不正确参数
‘81’	功能不被支持
‘82’	文件未找到
‘83’	记录未找到
‘84’	无足够的文件存储空间
‘85’	Lc 与 TLV 结构不一致
‘86’	不正确的参数 P1-P2
‘87’	Lc 与 P1-P2 不一致
‘88’	引用的数据未找到

9　APDU 指令

9.1　设备管理指令

9.1.1　概述

本部分 APDU 指令对应于 GM/T BBBB 中的设备管理类函数接口。

其中只有 SKF_SetLabel、SKF_GetDevInfo 两个函数与设备之间进行 APDU 指令的交互操作。

9.1.2 SetLabel(设置设备标签)

9.1.2.1 定义与范围

SetLabel 命令用于设置或修改设备的标签。

9.1.2.2 注意事项

标签数据长度应小于等于 32 字节,不能为空。

9.1.2.3 命令报文

表 12 SetLabel 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	02	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	标签数据的字节数
DATA	XX	XX...XX	标签数据
Le	—	—	不存在

9.1.2.4 命令报文数据域

命令报文数据域由待设置的设备标签组成。

9.1.2.5 响应报文数据域

没有响应报文数据。

9.1.2.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 13 SetLabel 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
65	81	写 EEPROM 失败
67	00	Lc 长度错误
69	82	不满足安全状态
6A	86	P1P2 参数不正确
6E	00	CLA 不正确

9.1.3 GetDevInfo(获取设备信息)

9.1.3.1 定义与范围

GetDevInfo 命令用于获取设备的一些特征信息,包括设备标签、厂商信息、支持的算法等。

9.1.3.2 注意事项

任何时候都可以执行此命令。

9.1.3.3 命令报文

表 14 GetDevInfo 命令报文编码

代 码	长度 (byte)	值 (Hex)	描 述
CLA	1	80	—
INS	1	04	—
P1	1	00	
P2	1	00	
Lc	—	—	不存在
DATA	—	—	不存在
Le	3	000000	期望返回全部设备信息

9.1.3.4 命令报文数据域

命令报文数据不存在。

9.1.3.5 响应报文数据域

响应报文数据为以下 cosDEVINFO 数据结构(需对此结构体以紧缩格式进行定义)的字节流编码。

```
typedef struct Struct_cosDEVINFO{
    VERSION     StructVersion;
    VERSION     SpecificationVersion;
    CHAR        Manufacturer[64];
    CHAR        Issuer[64];
    CHAR        Label[32];
    CHAR        SerialNumber[32];
    VERSION     HWVersion;
    VERSION     FirmwareVersion;
    ULONG       AlgSymCap;
    ULONG       AlgAsymCap;
    ULONG       AlgHashCap;
    ULONG       DevAuthAlgId;
    ULONG       TotalSpace;
```

```
    ULONG           FreeSpace;
    WORD            MaxApduDataLen;
    WORD            UserAuthMethod;
    WORD            DeviceType;
    BYTE            MaxContainerNum;
    BYTE            MaxCertNum;
    WORD            MaxFileNum;
    BYTE            Reserved[54];
}cosDEVINFO;

typedef struct Struct_Version{
    BYTE major;
    BYTE minor;
}VERSION;
```

表 15　设备信息描述

数据项	类型	意　义	备　注
StructVersion	VERSION	版本号	数据结构版本号，本结构的版本号为 1.0
SpecificationVersion	VERSION	版本号	本标准版本号，当前版本号为 1.0
Manufacturer	CHAR 数组	设备厂商信息	以‘\0’为结束符的 ASCII 字符串
Issuer	CHAR 数组	发行厂商信息	以‘\0’为结束符的 ASCII 字符串
Label	CHAR 数组	设备标签	以‘\0’为结束符的 ASCII 字符串
SerialNumber	CHAR 数组	序列号	以‘\0’为结束符的 ASCII 字符串
HWVersion	VERSION	设备硬件版本	
FirmwareVersion	VERSION	设备本身固件版本	
AlgSymCap	ULONG	分组密码算法标识	分组密码算法标识的编码规则为：从低位到高位，第 0 位到第 7 位按位表示分组密码算法工作模式，第 8 位到第 31 位按位表示分组密码算法。当多个分组算法同时存在时，可用“或”的形式表示。详细定义见 GM/T 0006
AlgAsymCap	ULONG	非对称密码算法标识	非对称密码算法标识的编码规则为：从低位到高位，第 0 位到第 7 位为 0，第 8 位到第 15 位按位表示非对称密码算法的算法协议，如果所表示的非对称算法没有相应的算法协议则为 0，第 16 位到第 31 位按位表示非对称密码算法类型。当多个非对称算法同时存在时，可用“或”的形式表示。详细定义见 GM/T 0006
AlgHashCap	ULONG	密码杂凑算法标识	密码杂凑算法标识的编码规则为：从低位到高位，第 0 位到第 7 位表示密码杂凑算法，第 8 位到第 31 位为 0。当多个密码杂凑算法同时存在时，可用“或”的形式表示。详细定义见 GM/T 0006。

表 15（续）

数据项	类型	意　义	备　注
DevAuthAlgId	ULONG	设备认证使用的分组密码算法标识	默认采用 SM1 算法进行设备认证操作。设备可以支持多种认证算法，各算法按照与 AlgSymCap 同样的方式进行编码
TotalSpace	ULONG	设备总空间大小	
FreeSpace	ULONG	用户可用空间大小	
MaxApduDataLen	WORD	设备支持的 APDU 命令数据域最大长度	
UserAuthMethod	WORD	用户认证方式	是 PIN 认证还是其他认证方式（如指纹、键盘输入等） 1：PIN 认证； 2：指纹认证； 3：通过设备自带的键盘输入 PIN 码； 其他值暂无定义
DeviceType	WORD	设备类型	返回设备的类型，如 IC 卡、普通 Key、显示按键 Key、指纹 Key 等。 1：IC 卡； 2：普通 USBKey； 3：显示按键 Key（不带键盘）； 4：显示按键 Key（带键盘，可用于输入 PIN 码）； 5：指纹 Key 其他值暂无定义
MaxContainerNum	BYTE	应用支持的最大容器数量	0 表示不限制
MaxCertNum	BYTE	应用支持的最大证书数量	0 表示不限制
MaxFileNum	WORD	应用支持的最大文件数	0 表示不限制
Reserved	ByteArray	保留字节	用于今后扩展使用

表 16　版本定义

数据项	类型	意义	备　注
major	BYTE	主版本号	主版本号和次版本号以“.”分隔，例如 Version 1.0，主版本号为 1，次版本号为 0；Version 2.10，主版本号为 2，次版本号为 10。
minor	BYTE	次版本号	

9.1.3.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 17　GetDevInfo 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
6A	86	P1P2 参数不正确
6C	00	Le 错误，Le 必须为 000000
6E	00	CLA 错误

9.2　访问控制指令

9.2.1　概述

访问控制指令主要完成设备认证、PIN 码管理和安全状态管理等操作。

9.2.2　DevAuth（设备认证）

9.2.2.1　定义与范围

DevAuth 命令是设备对应用程序的认证。

9.2.2.2　注意事项

执行此命令前，需要通过取随机数命令从设备获取 8 字节随机数。设备认证流程见 GM/T BBBB 8.2.3 节的设备认证部分。

9.2.2.3　命令报文

表 18　DevAuth 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	10	—
P1	1	00	—
P2	1	XX	认证算法标识，0x00 默认为 SM1 算法
Lc	3	0000XX	—
DATA	XX	XX...XX	加密的认证数据
Le	—	—	不存在

9.2.2.4　命令报文数据域

命令报文数据域由加密的认证数据组成。

表 19　P2 的定义

P2	定　义
0x00	SM1(设备必须支持,默认算法)
0x01	SSF33
0x02	SM4

9.2.2.5　响应报文数据域

没有响应报文数据。

9.2.2.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 20　DevAuth 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
63	CX	认证失败,还剩下 X 次重试机会
67	00	长度错误(Lc 域为空)
69	83	认证方法锁定
69	84	引用数据无效
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.2.3　ChangeDevAuthKey(修改设备认证密钥)

9.2.3.1　定义与范围

ChangeDevAuthKey 命令用于更改设备认证密钥。

9.2.3.2　注意事项

执行此命令前,需要通过取随机数命令获取 8 字节随机数。

9.2.3.3　命令报文

表 21　ChangeDevAuthKey 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	84	—
INS	1	12	—
P1	1	00	—

表 21（续）

代码	长度 （byte）	值 （Hex）	描　　述
P2	1	XX	认证算法标识，0x00 默认为 SM1 算法
Lc	3	000014	—
DATA	20	XX...XX	新设备认证密钥的密文数据（16 字节）＋报文认证码（MAC）数据元（4 字节）
Le	—	—	不存在

9.2.3.4　命令报文数据域

命令报文数据域由新设备认证密钥的密文数据和报文认证码（MAC）数据元组成，用于加密及 MAC 运算的密钥为原设备认证密钥。数据加密及 MAC 运算过程请参考附录 B：安全报文计算说明。

P2 的定义：

P2	定　　义
0x00	SM1（设备必须支持，默认算法）
0x01	SSF33
0x02	SM4

9.2.3.5　响应报文数据域

没有响应报文数据。

9.2.3.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 22　ChangeDevAuthKey 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
65	81	写 EEPROM 失败
67	00	长度错误（Lc 域为空）
69	83	设备认证密钥已经锁定
69	84	引用数据无效
63	CX	校验失败，X 为可重试次数（设备内在 MAC 校验失败时返回此错误代码）。
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.2.4　GetPinInfo（获取 PIN 信息）

9.2.4.1　定义与范围

GetPinInfo 命令是获取指定应用下的 PIN 码信息，包括最大重试次数、当前剩余重试次数，以及当

前 PIN 码是否为出厂默认 PIN 码。

9.2.4.2 注意事项

无。

9.2.4.3 命令报文

表 23 GetPinInfo 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	14	—
P1	1	00	—
P2	1	XX	PIN 类型(0x00 为管理员 PIN,0x01 为用户 PIN)
Lc	3	000002	应用 ID 长度
DATA	2	XXXX	应用 ID
Le	2	0003	最大重试次数(1 字节)+当前剩余重试次数(1 字节)+出厂默认 PIN 码状态(1 字节)

9.2.4.4 命令报文数据域

命令报文数据域为应用 ID。

9.2.4.5 响应报文数据域

响应报文数据由最大重试次数(1 字节)+当前剩余重试次数(1 字节)+出厂默认 PIN 码状态(1 字节)组成。出厂默认 PIN 码状态为‘01’表示建立应用后,PIN 码还没有修改过。

9.2.4.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 24 GetPinInfo 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6C	03	Le 长度错误,Le 应该为 000003
69	86	没有打开应用
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.2.5 ChangePin(修改 PIN)

9.2.5.1 定义与范围

ChangePin 命令用来修改指定应用的管理员 PIN 值或者用户 PIN 值。

9.2.5.2 注意事项

执行此命令前,需要预先打开应用,并通过取随机数命令获取 8 字节随机数。

9.2.5.3 命令报文

表 25 ChangePin 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	84	—
INS	1	16	—
P1	1	00	—
P2	1	XX	PIN 类型(0x00 为管理员 PIN,0x01 为用户 PIN)
Lc	3	0000XX	—
DATA	XX	XX...XX	应用 ID(2 字节)+ 加密的新 PIN + 报文认证码(MAC)数据元(4 字节)
Le	—	—	不存在

9.2.5.4 命令报文数据域

命令报文数据域由应用 ID、加密的新 PIN 和报文认证码(MAC)数据元组成。

加密以及 MAC 运算的密钥为原 PIN 经过摘要运算(采用 SHA1 算法)后的前 16 字节数据。

加密以及 MAC 计算采用 SM4 算法。

加密过程参见附录 B:安全报文计算过程。

9.2.5.5 响应报文数据域

没有响应报文数据。

9.2.5.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 26 ChangePin 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
65	81	写 EEPROM 失败
67	00	长度错误(Lc 域为空)
69	83	PIN 码已经被锁定
69	84	引用数据无效
63	CX	校验失败,X 为可重试次数(设备内在 MAC 校验失败时返回此错误代码)
6A	80	数据域不正确(新 PIN 码加密的补足错误)
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.2.6 VerifyPin（校验 PIN）

9.2.6.1 定义与范围

VerifyPin 命令是校验 PIN 码。校验成功后，会获得相应的权限，如果 PIN 码错误，会返回 PIN 码的重试次数，当重试次数为 0 时表示 PIN 码已经锁死。

9.2.6.2 注意事项

执行此命令前，需要预先打开应用，并通过取随机数命令获取 8 字节随机数。

9.2.6.3 命令报文

表 27 VerifyPin 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	18	—
P1	1	00	—
P2	1	XX	PIN 类型(0x00 为管理员 PIN，0x01 为用户 PIN)
Lc	3	000012	—
DATA	18	XX...XX	应用 ID(2 字节)＋ 加密的校验数据(16 字节)
Le	—	—	不存在

9.2.6.4 命令报文数据域

命令报文数据域为应用 ID 和使用加密密钥对 8 字节随机数进行加密后的结果组成。

加密密钥为 PIN 经过 HASH-SHA1 后的前 16 字节数据。

加密算法采用 SM4 算法。

加密过程参见附录 B:安全报文计算过程。

9.2.6.5 响应报文数据域

没有响应报文数据。

9.2.6.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 28 VerifyPin 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域为空)
63	CX	校验失败，X 为可重试次数

表 28（续）

SW1	SW2	意　义
69	83	认证方法锁定
69	84	引用数据无效
69	8A	没有打开应用
6A	86	参数 P1、P2 不正确

9.2.7 UnblockPin（解锁 PIN）

9.2.7.1 定义与范围

UnblockPin 命令是解锁锁死的用户 PIN 码。解锁后，用户 PIN 码被设置成新值，用户 PIN 码的重试次数也恢复到初始值。

9.2.7.2 注意事项

执行此命令前，需要预先打开应用，并通过取随机数命令获取 8 字节随机数。

9.2.7.3 命令报文

表 29　UnblockPin 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	84	—
INS	1	1A	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	—
DATA	XX	XX...XX	应用 ID(2 字节)＋ 加密的新用户 PIN ＋ 报文认证码(MAC)数据元(4 字节)
Le	—	—	不存在

9.2.7.4 命令报文数据域

命令报文数据域由应用 ID、加密的新用户 PIN 和报文认证码(MAC)数据元组成。

加密及线路保护密钥为管理员 PIN 经过 HASH-SHA1 后的前 16 字节数据。

加密以及 MAC 计算采用 SM4 算法。

加密过程参见附录 B:安全报文计算过程。

9.2.7.5 响应报文数据域

没有响应报文数据。

9.2.7.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 30 UnblockPin 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
65	81	写 EEPROM 失败
67	00	长度错误(Lc 域为空)
63	CX	校验失败,X 为可重试次数。当 X 为 0 时,管理员 PIN 锁死。
69	83	认证方法锁定
69	84	引用数据无效
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.2.8 ClearSecureState（清除应用安全状态）

9.2.8.1 定义与范围

ClearSecureState 命令用于清除指定应用的安全状态。安全状态至少包括 PIN 码认证状态。

9.2.8.2 注意事项

执行此命令前,需要先执行打开应用命令。

9.2.8.3 命令报文

表 31 ClearSecureState 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	1C	—
P1	1	00	—
P2	1	00	—
Lc	3	000002	应用 ID 长度
DATA	2	XXXX	应用 ID
Le	—	—	不存在

9.2.8.4 命令报文数据域

无命令报文数据域。

9.2.8.5 响应报文数据域

无响应报文数据。

9.2.8.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 32 ClearSecureState 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	Lc 长度错误
69	8A	没有打开应用
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.3 应用管理指令

9.3.1 概述

应用管理指令主要完成应用的创建、枚举、删除、打开、关闭等操作。

9.3.2 CreateApplication（创建应用）

9.3.2.1 定义与范围

CreateApplication 命令用于在设备上创建一个应用。

9.3.2.2 注意事项

需设备认证通过后才可以执行此命令。同一个设备中不允许存在相同名称的应用。

9.3.2.3 命令报文

表 33 CreateApplication 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	20	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	cosAPPLICATIONINFO 结构体的大小
DATA	XX	XXXXXX	cosAPPLICATIONINFO 结构体的字节流编码数据

9.3.2.4 命令报文数据域

命令报文数据为以下结构体(以紧缩格式表示)的字节流编码数据。

```
typedef struct Struct_cosAPPLICATIONINFO{
    CHAR    szApplicatinName[32];      //应用名称,不足 32 字节数据以 0x00 补全
    CHAR    szAdminPin[16];            //管理员口令,不足 16 字节数据以 0x00 补全
    ULONG   dwAdminPinRetryCount;      //管理员口令重试次数
    CHAR    szUserPin[16];             //用户口令,不足 16 字节数据以 0x00 补全
    ULONG   dwUserPinRetryCount;       //用户口令重试次数
    ULONG   dwCreateFileRights;        //在应用下创建文件的权限,权限信息的定义
                                       见//GM/T BBBB 中的权限类型定义。
    BYTE    byContainerNum;            //要求应用支持的容器数量
    BYTE    byCertNum;                 //要求应用支持的证书数量
    WORD    wFileNum;                  //要求应用支持的文件数量
} cosAPPLICATIONINFO;
```

9.3.2.5 响应报文数据域

无响应数据。

9.3.2.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 34 CreateApplication 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	Lc 长度错误(Lc 域给出的长度与数据结构体字节数据不一致)
69	82	安全状态不满足
6A	84	设备空间不足,无法创建更多应用
6A	86	P1P2 参数错误
6A	89	应用已经存在
6E	00	CLA 错误

9.3.3 EnumApplication(枚举应用)

9.3.3.1 定义与范围

EnumApplication 命令用于枚举设备上存在的所有应用。

9.3.3.2 注意事项

任何时候都可以执行此命令。

9.3.3.3 命令报文

表 35 EnumApplication 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	22	—
P1	1	00	—
P2	1	00	—
Lc	—	—	不存在
DATA	—	—	不存在
Le	3	000000	期望返回所有的应用名称

9.3.3.4 命令报文数据域

命令报文数据不存在。

9.3.3.5 响应报文数据域

返回应用名称列表。每个应用的名称以单个'\0'结束,以双'\0'表示列表的结束。

9.3.3.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 36 EnumApplication 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行,所有数据都已经返回
67	00	Lc 长度错误(Lc 域给出的长度与数据结构体字节数据不一致)
6A	86	P1P2 参数错误
6E	00	CLA 错误
6A	9E	还有更多数据需要上传,接口层需重新发送本条指令获取后续数据

9.3.4 DeleteApplication(删除应用)

9.3.4.1 定义与范围

DeleteApplication 命令用于删除设备上的一个应用。

9.3.4.2 注意事项

需设备认证通过后才可以执行此命令。

9.3.4.3 命令报文

表 37 DeleteApplication 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	24	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用名称长度(不含'\0'字符)
DATA	XX	XXXXXX	应用名称

9.3.4.4 命令报文数据域

命令报文数据为待删除应用的名称。

9.3.4.5 响应报文数据域

无响应数据。

9.3.4.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 38 DeleteApplication 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与 DATA 域数据长度不一致)
69	82	安全状态不满足
6A	8B	指定的应用不存在

9.3.5 OpenApplication（打开应用）

9.3.5.1 定义与范围

OpenApplication 命令用于打开设备上的一个应用。

9.3.5.2 注意事项

任何时候都可执行此命令。应用可被多次重复打开；设备厂商自行决定是否支持多应用的并发访问。

9.3.5.3 命令报文

表 39 OpenApplication 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	26	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用名称长度(不含'\0'字符)
DATA	XX	XXXXXX	应用名称
Le	2	00XX	期望返回应用属性信息的长度

9.3.5.4 命令报文数据域

命令报文数据为待打开应用的名称。

9.3.5.5 响应报文数据域

返回应用的属性信息。如应用权限、应用可支持的最大容器数据、最大证书个数、最大文件数量。

表 40 OpenApplication 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	dwCreateFileRights	ULONG	4	在该应用下创建文件和容器的权限
4	byMaxContainerNum	BYTE	1	指定应用可支持的最大容器数量
5	byMaxCertNum	BYTE	1	指定应用可支持的最大证书数量
6	wMaxFileNum	WORD	2	指定应用可支持的最大文件数量
8	wAppID	WORD	2	返回的应用 ID,用于标识已打开的应用,后续操作可通过此 ID 引用打开的应用

9.3.5.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 41 OpenApplication 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与 DATA 域数据长度不一致)
6A	8A	指定的应用已打开
6A	8B	应用不存在
6A	90	已有打开的应用,当前设备不支持同时打开多个应用

9.3.6 CloseApplication（关闭应用）

9.3.6.1 定义与范围

CloseApplication 命令用于关闭设备上的一个已打开应用。

9.3.6.2 注意事项

任何情况下都可以执行此命令。关闭应用只需调用一次，关闭应用不改变应用的安全状态。

9.3.6.3 命令报文

表 42 CloseApplication 命令报文编码

代码	长度（byte）	值（Hex）	描　　述
CLA	1	80	—
INS	1	28	—
P1	1	00	—
P2	1	00	—
Lc	3	000002	应用 ID 数据长度，为 2 字节
DATA	2	XXXX	应用 ID

9.3.6.4 命令报文数据域

命令报文数据为待关闭应用的 ID。

9.3.6.5 响应报文数据域

无响应数据。

9.3.6.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 43 CloseApplication 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误（Lc 域给出的长度与 DATA 域数据长度不一致）
6A	88	引用的应用不存在

9.4 文件管理指令

9.4.1 概述

文件管理指令用以满足用户扩展开发的需要，包括创建文件、删除文件、枚举文件、获取文件信息、文件读写等操作。本部分 APDU 指令对应于 GM/T BBBB 中的文件管理类函数接口。

9.4.2 CreateFile(创建文件)

9.4.2.1 定义与范围

CreateFile 命令用于在设备上的指定应用下创建一个文件。

9.4.2.2 注意事项

需具有应用指定的创建文件的权限才能创建文件。

9.4.2.3 命令报文

表 44 CreateFile 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	30	—
P1	1	XX	应用 ID 的高字节
P2	1	XX	应用 ID 的低字节
Lc	3	0000XX	FILEATTRIBUTE 结构体的大小
DATA	XX	XXXXXX	FILEATTRIBUTE 结构体的字节流编码数据

9.4.2.4 命令报文数据域

命令报文数据为 GM/T BBBB 中 FILEATTRIBUTE 结构体的字节流编码数据,具体见下表。

表 45 CreateFile 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	FileName	CHAR 数组	32	文件名,不足 32 字节则后面用 0 补位
32	FileSize	ULONG	4	文件大小
36	ReadRights	ULONG	4	读权限控制标识
40	WriteRights	ULONG	4	写权限控制标识

其中读写权限控制标识的具体含义见 GM/T BBBB 的“6.4.12 权限类型”一节。

9.4.2.5 响应报文数据域

无响应数据。

9.4.2.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 46　CreateFile 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误
6A	88	引用的应用不存在
69	82	安全状态不满足
6A	84	设备空间不足,无法创建更多文件
6A	92	同名文件已存在

9.4.3　DeleteFile(删除文件)

9.4.3.1　定义与范围

DeleteFile 命令用于删除设备上的一个文件。

9.4.3.2　注意事项

需具有应用指定的创建文件的权限才能删除文件。该指令执行成功后,该文件写入的所有信息将丢失。

9.4.3.3　命令报文

表 47　DeleteFile 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	32	—
P1	1	XX	应用 ID 的高字节
P2	1	XX	应用 ID 的低字节
Lc	3	0000XX	文件名长度
DATA	XX	XXXXXX	文件名

9.4.3.4　命令报文数据域

命令报文数据域为文件名,不包含结尾的'\0'。

9.4.3.5　响应报文数据域

无响应数据。

9.4.3.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 48　DeleteFile 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误
6A	88	引用的应用不存在
69	82	安全状态不满足
65	81	写 EEPROM 失败
6A	93	文件不存在

9.4.4　EnumFiles(枚举文件)

9.4.4.1　定义与范围

EnumFiles 命令用于枚举设备上指定应用下存在的所有文件。

9.4.4.2　注意事项

无。

9.4.4.3　命令报文

表 49　EnumFiles 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	34	—
P1	1	XX	应用 ID 的高字节
P2	1	XX	应用 ID 的低字节
Lc	—	—	不存在
DATA	—	—	不存在
Le	3	000000	期望返回所有的文件名称

9.4.4.4　命令报文数据域

命令报文数据不存在。

9.4.4.5　响应报文数据域

返回文件名称列表。每个文件的名称以单个'\0'结束,以双'\0'表示列表的结束。

9.4.4.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 50　EnumFiles 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
6A	88	应用不存在
69	82	安全状态不满足
6A	9E	还有更多数据需要上传，接口层需重新发送本条指令获取后续数据

9.4.5　GetFileInfo(获取文件信息)

9.4.5.1　定义与范围

GetFileInfo 命令用于获取指定应用目录下指定文件的属性信息。

9.4.5.2　注意事项

无。

9.4.5.3　命令报文

表 51　GetFileInfo 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	36	—
P1	1	XX	应用 ID 的高字节
P2	1	XX	应用 ID 的低字节
Lc	3	0000XX	文件名长度
DATA	XX	XX..XX	文件名
Le	2	00XX	期望返回的数据字节

9.4.5.4　命令报文数据域

命令报文数据为文件名称，不包含结尾的'\0'。

9.4.5.5　响应报文数据域

响应报文数据的编码结构见下表。

表 52　GetFileInfo 响应报文编码

偏移	数据域	数据类型	字节长度	说　明
0	FileSize	ULONG	4	文件大小
4	ReadRights	ULONG	4	读权限控制标识
8	WriteRights	ULONG	4	写权限控制标识

其中读写权限控制标识的具体含义见 GM/T BBBB 的“6.4.12 权限类型”一节。

9.4.5.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 53 GetFileInfo 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Le 域给出的长度与数据结构体字节数据不一致)
6A	88	应用不存在
6A	93	文件不存在
69	82	安全状态不满足

9.4.6 ReadFile（读文件）

9.4.6.1 定义与范围

ReadFile 命令用于从指定文件的指定开始位置读取指定长度的数据。

9.4.6.2 注意事项

无。

9.4.6.3 命令报文

表 54 ReadFile 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	38	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	报文数据长度
DATA	XX	XXXXXX	报文数据
Le	2	XXXX	期望读取的数据长度,0 表示读取所有数据

9.4.6.4 命令报文数据域

命令报文编码如下表所示：

表 55 ReadFile 命令报文编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wOffset	WORD	2	数据读取的偏移量
4	wReadLen	WORD	2	期望读取数据的长度(若此值为 0,表示返回所有数据)
6	wFileNameLen	WORD	2	文件名称长度,不含结尾的'\0'字符
8	chFileName	Char array	wFileNameLen	文件名称

9.4.6.5 响应报文数据域

响应报文数据的长度由 Le 的值决定(Le 的值应该与命令报文编码中的 wReadLen 相等)。如果 Le 超过可读取的实际数据长度,则返回实际长度的数据。

9.4.6.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 56 ReadFile 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
69	82	权限不满足
6A	93	指定的文件不存在
6B	00	偏移值超出文件长度
6E	01	待返回数据长度超出最大缓冲区长度

9.4.7 WriteFile(写文件)

9.4.7.1 定义与范围

WriteFile 命令用于向指定应用的指定文件的指定位置写入指定长度的数据

9.4.7.2 注意事项

需满足文件的写入权限。

9.4.7.3 命令报文

表 57 WriteFile 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	3A	—

表 57（续）

代码	长度 （byte）	值 （Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	00XXXX	XXXXXX	报文数据
Le	—	—	不存在

9.4.7.4　命令报文数据域

命令报文数据为应用 ID、文件名、偏移地址及待写入的数据。

命令报文编码如下表所示：

表 58　WriteFile 命令报文编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wOffset	WORD	2	数据写入的偏移量
4	wFileNameLen	WORD	2	文件名称长度，不含结尾的‘\0’字符
6	chFileName	Char array	wFileNameLen	文件名称
6＋wFileNameLen	wDataLen	WORD	2	待写入数据的长度
8＋wFileNameLen	pbData	ByteArray	wDataLen	待写入数据

9.4.7.5　响应报文数据域

无响应报文数据。

9.4.7.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 59　WriteFile 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误（偏移量和 Lc 之和超过文件长度）
69	82	权限不满足
6A	93	指定的文件不存在
6B	00	偏移值超出文件长度

9.5 容器管理指令

9.5.1 概述

容器管理指令用于满足各种不同容器的管理，包括创建、删除、枚举、打开、关闭容器以及证书导入、导出等操作。

9.5.2 CreateContainer(创建容器)

9.5.2.1 定义与范围

CreateContainer 命令用于在特定的应用上创建一个容器。

9.5.2.2 注意事项

需该应用的用户认证通过后才可以执行此命令，容器名称的最大长度不超过 34 字节。在特定的应用下创建容器。同一应用下不允许创建同名的容器。

9.5.2.3 命令报文

表 60 CreateContainer 命令报文编码

代码	长度(byte)	值(Hex)	描　　述
CLA	1	80	—
INS	1	40	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器名称的字节数
DATA	XX	XX…XX	应用 ID(2 个字节，高端存储)和容器名称
LE	2	0002	返回容器 ID 值

9.5.2.4 命令报文数据域

命令报文数据由待创建的容器名称组成。

9.5.2.5 响应报文数据域

响应报文为返回的容器 ID(2 字节)。

9.5.2.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 61 CreateContainer 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)

表 61（续）

SW1	SW2	意　义
69	82	安全状态不满足
6A	84	设备空间不足，无法创建更多容器
65	81	写 EEPROM 失败
6E	02	指定的容器已存在
6A	88	引用的应用未找到

9.5.3 OpenContainer（打开容器）

9.5.3.1 定义与范围

OpenContainer 命令用于在设备的特定应用上打开一个容器。

9.5.3.2 注意事项

打开该应用的前提下，可以执行此命令。允许重复打开容器。

9.5.3.3 命令报文

表 62 OpenContainer 命令报文编码

代码	长度（byte）	值（Hex）	描　述
CLA	1	80	—
INS	1	42	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器名称的字节数
DATA	XX	XX…XX	应用 ID（2 个字节，高端存储）和容器名称值
Le	2	0002	返回容器 ID 值（2 字节）

9.5.3.4 命令报文数据域

命令报文数据为应用 ID 和待打开容器名称。

9.5.3.5 响应报文数据域

响应报文为返回的容器 ID（2 字节）。

9.5.3.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 63　OpenContainer 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用未找到
6A	91	指定的容器不存在

9.5.4　CloseContainer(关闭容器)

9.5.4.1　定义与范围

CloseContainer 命令用于在指定应用上关闭一个容器。

9.5.4.2　注意事项

已经打开该应用并且已经打开该容器的情况下,可以成功执行此命令。关闭容器只需调用一次。

9.5.4.3　命令报文

表 64　CloseContainer 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	44	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器 ID 的字节数
DATA	XX	XX…XX	应用 ID(2 个字节,高端存储)和容器 ID(2 个字节,高端存储)

9.5.4.4　命令报文数据域

命令报文数据由应用 ID 和待关闭的容器 ID 组成。

9.5.4.5　响应报文数据域

无响应数据。

9.5.4.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 65　CloseContainer 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	8B	引用的应用未找到
6A	94	引用的容器未找到

9.5.5　EnumContainer（枚举容器）

9.5.5.1　定义与范围

EnumContainer 命令用于枚举特定应用中存在的所有容器。

9.5.5.2　注意事项

已经打开该应用的情况下,可以成功执行此命令。

9.5.5.3　命令报文

表 66　EnumContainer 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	46	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 字节数
DATA	XX	XX..XX	应用 ID(2 个字节,高端存储)
Le	—	—	不存在

9.5.5.4　命令报文数据域

命令报文数据不存在。

9.5.5.5　响应报文数据域

响应报文数据是返回的容器名称列表。每个应用的名称以单个‘\0’结束,以双‘\0’表示列表的结束。

9.5.5.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 67　EnumContainer 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Le 域给出的长度与数据字节数不一致)
6A	88	引用的应用未找到
6A	9E	还有更多数据需要上传,接口层需重新发送本条指令获取后续数据

9.5.6　DeleteContainer(删除容器)

9.5.6.1　定义与范围

DeleteContainer 命令用于删除特定应用上指定的容器。

9.5.6.2　注意事项

已经打开该应用的情况下,需用户认证该应用下的用户密码后,可以成功执行此命令。

9.5.6.3　命令报文

表 68　DeleteContainer 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	48	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器名称的字节数
DATA	XX	XX…XX	应用 ID(2 个字节,高端存储)和容器名称值

9.5.6.4　命令报文数据域

命令报文数据由应用 ID 和待删除的容器名称组成。

9.5.6.5　响应报文数据域

无响应数据。

9.5.6.6　响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 69　DeleteContainer 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	引用的应用未找到
6A	91	指定的容器未找到

9.5.7　GetContainerInfo(获取容器信息)

9.5.7.1　定义与范围

GetContainerInfo 命令用于获取特定应用下的特定容器的相关信息。

9.5.7.2　注意事项

已经打开应用的情况下都可以执行此命令。

9.5.7.3　命令报文

表 70　GetContainerInfo 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	4A	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器名称的字节数
DATA	XX	XX..XX	应用 ID(2 个字节,高端存储)和容器名称值
Le	2	000B	

9.5.7.4　命令报文数据域

本命令无报文数据。

9.5.7.5　响应报文数据域

表 71　GetContainerInfo 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	ContainerType	BYTE	1	容器类型,0 表示未定、尚未分配类型或者为空容器,1 表示为 RSA 容器,2 表示为 ECC 容器

表 71（续）

偏移	数据域	数据类型	字节长度	说　明
1	ulSignKeyLen	ULONG	4	签名密钥对的位长度，未生成则此值为 0
5	ulExchgKeyLen	ULONG	4	加密密钥对的位长度，未导入则此值为 0
9	bSignCertFlag	BYTE	1	是否存在签名证书的标志字节。1 表示存在，0 表示不存在，其他值无效
10	bExchgCertFlag	BYTE	1	是否存在加密证书的标志字节。1 表示存在，0 表示不存在，其他值无效

9.5.7.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 72　GetContainerInfo 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	88	引用的应用未找到
6A	91	指定的容器未找到

9.5.8　ImportCertificate（导入数字证书）

9.5.8.1　定义与范围

ImportCertificate 命令用于向当前容器内导入指定类型的数字证书。

9.5.8.2　注意事项

用户认证通过后才可以执行此命令。

9.5.8.3　命令报文

表 73　ImportCertificate 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	4C	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	应用 ID、容器 ID 和证书数据报文数据总长度
DATA	XXXX	XX..XX	应用 ID、容器 ID 和数字证书信息

9.5.8.4 命令报文证书数据

表 74 ImportCertificate 命令报文证书数据

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	bCertType	BYTE	1	证书类型，1 表示为签名证书，0 表示加密证书
5	ulCertLen	ULONG	4	证书数据长度
9	pbCert	ByteArray	ulCertLen	证书数据

9.5.8.5 响应报文数据域

本命令无响应报文数据

9.5.8.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 75 ImportCertificate 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用不存在
6A	94	应用的容器不存在
6A	95	容器中没有对应的密钥对

9.5.9 ExportCertificate(导出数字证书)

9.5.9.1 定义与范围

ExportCertificate 命令用于将当前容器内指定类型的数字证书导出。

9.5.9.2 注意事项

应用、容器打开的情况下可以执行此命令。

9.5.9.3 命令报文

表 76 ExportCertificate 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	4E	—

表 76（续）

代码	长度（byte）	值（Hex）	描　　述
P1	1	XX	希望导出的证书类型
P2	1	00	—
Lc	3	0000XX	应用 ID 和容器 ID 的字节数
DATA	XX	XX..XX	应用 ID、容器 ID
Le	2	0000	

9.5.9.4　命令报文数据域

本命令无报文数据。

9.5.9.5　响应报文数据域

表 77　ExportCertificate 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulCertLen	ULONG	4	证书数据长度
4	pbCert	ByteArray	ulCertLen	证书数据

9.5.9.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 78　ExportCertificate 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6A	94	引用的容器不存在
6A	88	引用的应用不存在
6A	96	指定类型的证书不存在

9.6　密码服务指令

9.6.1　概述

密码服务指令提供对称算法运算、非对称算法运算、密码杂凑运算、密钥管理、消息鉴别码计算等功能。

9.6.2　GenRandom（生成随机数）

9.6.2.1　定义与范围

GenRandom 命令用于产生指定长度的随机数。

9.6.2.2 注意事项

任何时候都可以执行此命令。

9.6.2.3 命令报文

表 79 GenRandom 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	50	—
P1	1	00	—
P2	1	00	—
Lc	—	—	不存在
DATA	—	—	不存在
Le	3	00XXXX	指定长度的随机数

9.6.2.4 命令报文数据域

无命令报文数据域。

9.6.2.5 响应报文数据域

响应报文数据域为随机数。

9.6.2.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 80 GenRandom 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	Lc 长度错误
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.6.3 GenExtRSAKey（生成外部 RSA 密钥对）

9.6.3.1 定义与范围

GenExtRSAKey 命令是由设备生成 RSA 密钥对并明文输出。

9.6.3.2 注意事项

任何时候都可以执行此命令。

9.6.3.3 命令报文

表 81 GenExtRSAKey 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	52	—
P1	1	00	
P2	1	00	—
Lc	3	000002	—
DATA	2	XXXX	RSA 模长 BitLen
Le	2	XXXX	RSA 编码数据长度

9.6.3.4 命令报文数据域

命令报文数据域为空。

9.6.3.5 响应报文数据域

响应报文数据域为生成密钥对的私钥数据。

表 82 GenExtRSAKey 响应报文编码

偏移	数据域	数据类型	字节长度	说　　明
0	Modulus	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节，BitLen 为 RSA 模长。
BitLen/8	PublicExponent	BYTE 数组	4	一般为 0x00010001
4＋ BitLen/8	PrivateExponent	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节
4＋ BitLen/4	Prime1	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
4＋ 5 * BitLen/16	Prime2	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
4＋ 6 * BitLen/16	Prime1Exponent	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
4＋ 7 * BitLen/16	Prime2Exponent	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
4＋ 8 * BitLen/16	Coefficient	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节

9.6.3.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 83 GenExtRSAKey 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	Lc 长度错误

表 83（续）

SW1	SW2	意　义
6A	81	功能不支持
6A	86	P1P2 参数错误
6E	00	CLA 错误

9.6.4 GenRSAKeyPair（生成 RSA 签名密钥对）

9.6.4.1 定义与范围

GenKeyPair 命令是在指定的应用和容器中生成 RSA 签名密钥对并输出签名公钥。

9.6.4.2 注意事项

需通过用户权限验证。

9.6.4.3 命令报文

表 84 GenRSAKeyPair 命令报文编码

代码	长度（byte）	值（Hex）	描　述
CLA	1	80	—
INS	1	54	—
P1	1	00	—
P2	1	00	—
Lc	3	000006	—
DATA	6	XXXX	应用 ID(2 字节)＋容器 ID(2 字节)＋RSA 模长(2 字节)
Le	2	XXXX	

9.6.4.4 命令报文数据域

报文数据域为应用 ID(2 字节)＋容器 ID(2 字节)＋RSA 模长(2 字节)。

9.6.4.5 响应报文数据域

响应报文数据域为签名密钥对的公钥数据。

表 85 GenRSAKeyPair 响应报文编码

偏移	数据域	数据类型	字节长度	说　明
0	Modulus	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节，BitLen 为 RSA 模长
BitLen/8	PublicExponent	BYTE 数组	4	一般为 0x00010001

9.6.4.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 86 GenRSAKeyPair 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
65	81	写 EEPROM 失败
67	00	Lc 长度错误
69	82	不满足安全状态
6A	81	功能不支持
6A	84	空间不足
6A	86	P1P2 参数不正确
6E	00	CLA 错误
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.5 ImportRSAKeyPair（导入 RSA 加密密钥对）

9.6.5.1 定义与范围

ImportRSAKeyPair 命令在指定应用的容器中导入 RSA 加密公私钥对。

9.6.5.2 注意事项

需通过用户权限验证。

9.6.5.3 命令报文

表 87 ImportRSAKeyPair 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	56	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	—
DATA	XXXX	XX…XX	详见命令报文数据域
Le	—	—	不存在

9.6.5.4 命令报文数据域

命令报文数据域由对称算法密钥标识、使用指定签名公钥保护的对称算法密钥和对称算法密钥保护的 RSA 加密私钥组成。私钥的格式遵循 PKCS #1 v2.1：RSA Cryptography Standard 中的私钥格式定义。

表 88 ImportRSAKeyPair 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulSymAlgID	ULONG	4	用于加密待导入加密密钥对私钥的会话密钥算法标识
8	ulWrappedKeyLen	ULONG	4	签名公钥加密后的会话密钥密文长度
12	pbWrappedKey	ByteArray	ulWrapped KeyLen	签名公钥加密的会话密钥密文
	ulBits	ULONG	4	待导入加密密钥对的密钥位长度
	ulEncryptedDataLen	ULONG	4	待导入加密密钥对私钥密文长度(字节为单位)
	pbEncryptedData	ByteArray	ulEncrypted DataLen	待导入加密密钥对的密文,加密密钥对私钥原文格式遵循 PKCS #1 v2.1：RSA Cryptography Standard 中的私钥格式定义

9.6.5.5 响应报文数据域

无响应报文数据域。

9.6.5.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 89 ImportRSAKeyPair 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域为空)
69	82	不满足安全状态
6A	84	空间不足
6A	94	引用的容器不存在
6A	88	引用的应用不存在

9.6.6 RSASignData（RSA 签名）

9.6.6.1 定义与范围

RSASignData 命令是使用指定签名私钥,对指定数据进行数字签名。签名后的结果输出。

9.6.6.2 注意事项

需通过用户权限验证。

9.6.6.3 命令报文

表 90 RSASignData 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	58	—
P1	1	XX	1:输入为待签名数据的原文,需在设备内进行摘要运算和PKCS1 编码处理 2:输入为原文摘要后的数据,若数据长度与公钥模长相等,则直接进行签名;其他情况则先进行 PKCS1 编码,然后签名
P2	1	XX	摘要算法标识: 1:SM3; 2:SHA1; 3:SHA256; 其他值暂不定义
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX…XX	应用 ID + 容器 ID + 待签名数据
Le	2	XXXX	签名结果长度

9.6.6.4 命令报文数据域

命令报文数据域为应用 ID、容器 ID 和待签名数据。

9.6.6.5 响应报文数据域

响应报文数据域为签名结果。

9.6.6.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 91 RSASignData 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域为空)
69	82	不满足安全状态
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	95	容器中对应密钥对不存在

9.6.7 RSAVerify(RSA 验签)

9.6.7.1 定义与范围

RSAVerify 命令用 RSA 公钥(从外部输入)对数据进行签名验证。

9.6.7.2 注意事项

任何时候都可以执行此命令。

9.6.7.3 命令报文

表 92 RSAVerify 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	5E	—
P1	1	00	1:待验证数据为原文 2:待验证数据为原文摘要后的数据,若数据长度与公钥模长相等,则表示数据已经过 PKCS1 编码处理
P2	1	00	摘要算法标识: 1:SM3 2:SHA1 3:SHA256 其他值暂不定义
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	公钥、待验签数据和签名数据
Le	—	—	无

9.6.7.4 命令报文数据域

表 93 RSAVerify 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	BitLen	ULONG	4	输入公钥的密钥位长度
4	Modulus	Byte Array	BitLen/8	实际长度为 BitLen/8 字节,BitLen 为 RSA 模长
4+BitLen/8	PublicExponent	Byte Array	4	公钥模指数,一般为 0x00010001
8+BitLen/8	ulDataLen	ULONG	4	待验证签名的数据长度
12+BitLen/8	pbData	Byte Array	ulDataLen	待验证签名的数据
12+BitLen/8+ ulDataLen	ulSignLen	WORD	2	签名值长度,必须为公钥模长
14+BitLen/8+ ulDataLen	pbSignature	Byte Array	ulSignLen	待验证的签名值

9.6.7.5 响应报文数据域

本命令没有响应报文数据。

9.6.7.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 94 RSAVerify 命令响应状态码

SW1	SW2	意　义
90	00	正确执行，验签成功
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	98	正确执行，但验证签名失败

9.6.8 RSAExportSessionKey（RSA 生成并导出会话密钥）

9.6.8.1 定义与范围

RSAExportSessionKey 命令是生成会话密钥并用外部公钥加密输出。

9.6.8.2 注意事项

需要打开应用和容器。

9.6.8.3 命令报文

表 95 RSAExportSessionKey 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	5A	—
P1	1	00	
P2	1	00	
Lc	3	00XXXX	—
DATA	XXXX	XX…XX	详见命令报文数据域
Le	2	XXXX	加密的会话密钥值密文长度

9.6.8.4 命令报文数据域

命令报文数据域如下表所示。

表 96　RSAExportSessionKey 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulAlgId	ULONG	4	会话密钥算法标识
8	BitLen	ULONG	4	加密密钥对的密钥位长度
12	Modulus	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节
12＋ BitLen/8	PublicExponent	BYTE 数组	4	一般为 0x00010001

9.6.8.5　响应报文数据域

响应报文数据域编码如下表所示，会话密钥密文需按照 PKCS＃1v1.5 要求封装。

表 97　RSAExportSessionKey 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wSessionKeyID	WORD	2	生成的会话密钥 ID
2	pbData	ByteArray	BitLen/8	会话密钥密文

9.6.8.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 98　RSAExportSessionKey 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域为空)
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.9　RSAExportSessionKeyEx（RSA 导出会话密钥）

9.6.9.1　定义与范围

RSAExportSessionKeyEx 命令利用外部公钥加密输出设备中的会话密钥。

9.6.9.2　注意事项

需要打开应用和容器。该函数用于在 RSAExportSessionKey 或 ECCExportSessionKey 函数执行后继续使用其他 RSA 公钥导出会话密钥。

9.6.9.3 命令报文

表 99 RSAExportSessionKeyEx 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	5C	—
P1	1	00	
P2	1	00	
Lc	3	00XXXX	—
DATA	XXXX	XX…XX	详见命令报文数据域
Le	2	XXXX	加密的会话密钥值密文

9.6.9.4 命令报文数据域

命令报文数据域如下表所示。

表 100 RSAExportSessionKeyEx 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	wSessionKeyID	WORD	2	会话密钥 ID
6	BitLen	ULONG	4	加密密钥对的密钥位长度
10	Modulus	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节
10＋ BitLen/8	PublicExponent	BYTE 数组	4	一般为 0x00010001

9.6.9.5 响应报文数据域

响应报文数据域编码如下表所示，会话密钥密文需按照 PKCS＃1v1.5 要求封装。

表 101 RSAExportSessionKeyEx 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	pbData	ByteArray	BitLen/8	会话密钥密文

9.6.9.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 102　RSAExportSessionKeyEx 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域为空)
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	9F	引用的密钥未找到

9.6.10　ExtRSAPubKeyOperation(RSA 外来公钥运算)

9.6.10.1　定义与范围

ExtRSAPubKeyOperation 命令使用外部传入的 RSA 公钥对输入数据做公钥运算并输出结果。

9.6.10.2　注意事项

任何时候都可以执行此命令。

9.6.10.3　命令报文

表 103　ExtRSAPubKeyOperation 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	60	—
P1	1	00	
P2	1	00	
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	公钥、待运算数据
Le	2	XXXX	输出数据长度(为公钥模长的字节数)

9.6.10.4　命令报文数据域

表 104　ExtRSAPubKeyOperation 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	BitLen	ULONG	4	输入公钥的密钥位长度
4	Modulus	Byte Array	BitLen/8	实际长度为 BitLen/8 字节,BitLen 为 RSA 模长
4+BitLen/8	PublicExponent	Byte Array	4	公钥模指数,一般为 0x00010001
8+BitLen/8	ulInputLen	ULONG	4	待运算数据的长度,必须为公钥模长
12+BitLen/8	pbInput	Byte Array	ulDataLen	待运算数据

9.6.10.5 响应报文数据域

本命令返回公钥运算的结果。

9.6.10.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 105 ExtRSAPubKeyOperation 命令响应状态码

SW1	SW2	意　义
90	00	正确执行，公钥运算成功
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	98	公钥运算失败

9.6.11 ExtRSAPriKeyOperation（RSA 外来私钥运算）

9.6.11.1 定义与范围

ExtRSAPriKeyOperation 命令直接使用外部传入的 RSA 私钥对输入数据做私钥运算并输出结果。

9.6.11.2 注意事项

任何时候都可以执行此命令。

9.6.11.3 命令报文

表 106 ExtRSAPriKeyOperation 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	62	—
P1	1	00	
P2	1	00	
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	私钥、待运算数据
Le	2	XXXX	输出数据长度（为公钥模长的字节数）

9.6.11.4 命令报文数据域

表 107 ExtRSAPriKeyOperation 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	BitLen	ULONG	4	输入私钥的密钥位长度
4	Modulus	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节，BitLen 为 RSA 模长

表 107（续）

偏移	数据域	数据类型	字节长度	说　明
4+BitLen/8	PublicExponent	BYTE 数组	4	一般为 0x00010001
8+ BitLen/8	PrivateExponent	BYTE 数组	BitLen/8	实际长度为 BitLen/8 字节
8+ BitLen/4	Prime1	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
8+ 5 * BitLen/16	Prime2	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
8+ 6 * BitLen/16	Prime1Exponent	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
8+ 7 * BitLen/16	Prime2Exponent	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
8+ 8 * BitLen/16	Coefficient	BYTE 数组	BitLen/16	实际长度为 BitLen/16 字节
8+ 9 * BitLen/16	ulInputLen	ULONG	4	待运算数据的长度，必须为公钥模长
12+ 9 * BitLen/16	pbInput	Byte Array	ulDataLen	待运算数据

9.6.11.5 响应报文数据域

本命令返回私钥运算的结果，数据长度为公钥模长字节数。

9.6.11.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 108 ExtRSAPriKeyOperation 命令响应状态码

SW1	SW2	意　义
90	00	正确执行，私钥运算成功
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	98	私钥运算失败

9.6.12 GenECCKeyPair（生成 ECC 签名密钥对）

9.6.12.1 定义与范围

GenECCKeyPair 命令用于在指定应用的当前容器中生成 ECC 签名密钥对并输出签名公钥。

9.6.12.2 注意事项

需用户认证通过后才可以执行此命令。

9.6.12.3 命令报文

表 109 GenECCKeyPair 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	70	—

表 109（续）

代码	长度 （byte）	值 （Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	000008	DATA 域字节数为 8
DATA	08	XXXXXX	命令报文数据
Le	2	XXXX	期望返回的公钥数据字节长度

9.6.12.4　命令报文数据域

命令报文各数据域如以下表格所示进行编码。

表 110　GenECCKeyPair 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulBits	ULONG	4	期望生成密钥对的位长度

9.6.12.5　响应报文数据域

GM/T BBBB 中 ECCPUBLICKEYBLOB 结构的 XCoordinate 和 YCoordinate 数值。其编码格式如下表。

表 111　GenECCKeyPair 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	XCoordinate	Byte Array	ulBits/8	ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
ulBits/8	YCoordinate	Byte Array	ulBits/8	ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标

9.6.12.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 112　GenECCKeyPair 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）

表 112（续）

SW1	SW2	意　义
69	82	安全状态不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.13 ImportECCKeyPair(导入 ECC 加密密钥对)

9.6.13.1 定义与范围

ImportECCKeyPair 命令用于在指定应用的指定容器中导入 ECC 加密密钥对。

9.6.13.2 注意事项

需用户认证通过后才可以执行此命令。

9.6.13.3 命令报文

表 113 ImportECCKeyPair 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	72	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	导入的加密密钥对数据
Le	—	—	无

9.6.13.4 命令报文数据域

命令报文数据为经对称密钥加密后的待导入加密密钥对数据。

以下为相关数据结构定义。

```
typedef struct SKF_ENVELOPEDKEYBLOB{
    ULONG     ulAsymmAlgID;       // 保护对称密钥的非对称算法标识
    ULONG     ulSymmAlgID;        // 用于加密待导入 ECC 密钥对的对称算法标识,限定
                                  // 为采用 ECB 模式对密钥对数据进行加密
    ECCCIPHERBLOB ECCCipherBlob;// 用保护公钥加密的对称密钥密文。
    ECCPUBLICKEYBLOB PubKey;    // 待导入加密密钥对的公钥
    BYTE cbEncryptedPriKey[64];     // 待导入加密密钥对私钥的密文
}ENVELOPEDKEYBLOB, * PENVELOPEDKEYBLOB;
```

```
typedef struct Struct_ECCCIPHERBLOB{
    //SM2 加密算法输出密文中 C1 的 X 坐标值,其位长度与设备中签名公钥的位长度相等
    BYTE XCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    //SM2 加密算法输出密文中 C1 的 Y 坐标值,其位长度与设备中签名公钥的位长度相等
    BYTE YCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    //SM2 加密算法输出密文中的参数 C3
    BYTE HASH[32];
    //SM2 加密算法输出密文中参数 C2 的长度
    ULONG CipherLen;
    //SM2 加密算法输出密文中参数 C2 的值
    BYTE Cipher[1];
} ECCCIPHERBLOB, *PECCCIPHERBLOB;

typedef struct Struct_ECCPUBLICKEYBLOB{
    ULONG       BitLen;
    BYTE        XCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE        YCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
}ECCPUBLICKEYBLOB, *PECCPUBLICKEYBLOB;

typedef struct Struct_ECCPRIVATEKEYBLOB{
    ULONG       BitLen;
    BYTE        PrivateKey[ECC_MAX_XCOORDINATE_BITS_LEN/8];
}ECCPRIVATEKEYBLOB, *PECCPRIVATEKEYBLOB;
```

表 114　ImportECCKeyPair 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulAsymmAlgID	ULONG	4	用于保护对称密钥的非对称算法标识
8	ulSymmAlgID	ULONG	4	用于加密待导入加密密钥对的会话密钥算法标识
12	ulC1Bits	ULONG	4	ECCCIPHERBLOB 结构体中 X 坐标和 Y 坐标的位长度,其值应与设备中签名公钥的位长度相等
16	C1XCoordinate	Byte Array	ulC1Bits /8	ECCCIPHERBLOB 结构体中的 X 坐标
16+ulC1Bits/8	C1YCoordinate	Byte Array	ulC1Bits /8	ECCCIPHERBLOB 结构体中的 Y 坐标
16+ulC1Bits/4	HASH	Byte Array	32	ECCCIPHERBLOB 结构体中的 Hash 杂凑值
48+ulC1Bits/4	CipherLen	ULONG	4	ECCCIPHERBLOB 结构体中的密文数据长度
52+ulC1Bits/4	Cipher	Byte Array	CipherLen	ECCCIPHERBLOB 结构体中的密文数据,为会话密钥密文。

表 114（续）

偏移	数据域	数据类型	字节长度	说　明
	ulBits	ULONG	4	待导入加密密钥对的密钥位长度
	importedXCoordinate	Byte Array	ulBits/8	待导入加密密钥对公钥中的 X 坐标
	importedYCoordinate	Byte Array	ulBits/8	待导入加密密钥对公钥中的 Y 坐标
	ulEncryptedPriKeyLen	ULONG	4	待导入加密密钥对私钥密文长度(字节为单位)
	cbEncryptedPriKey	Byte Array	ulEncrypted PriKeyLen	待导入加密密钥对私钥的密文,加密私钥的原文为 ECCPRIVATEKEYBLOB 结构中的 PrivateKey 数据的有效部分。

9.6.13.5　响应报文数据域

本命令没有响应报文数据。

9.6.13.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 115　ImportECCKeyPair 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	引用的的应用不存在
6A	94	引用的的容器不存在
6A	97	数据写入失败

9.6.14　ECCSignData(ECC 签名)

9.6.14.1　定义与范围

ECCSignData 命令采用指定容器中 ECC 签名私钥对输入数据进行签名并返回签名数据。

9.6.14.2　注意事项

需用户认证通过后才可以执行此命令。

9.6.14.3　命令报文

表 116　ECCSignData 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	74	—

表 116（续）

代码	长度 (byte)	值 (Hex)	描　　述
P1	1	XX	1：输入为待签名数据的原文。当使用 SM2 算法时，需在设备内进行 SM2 预处理，然后对预处理结果进行签名操作 2：输入的待签名数据为原始数据的杂凑值。当使用 SM2 算法时，该待签名数据为原始数据经过 SM2 签名预处理的结果（SM2 的预处理过程见 GM/T AAAA） 其他值暂不支持
P2	1	00	—
Lc	3	00XXXX	命令报文数据长度
DATA	XXXX	XX..XX	P1＝1 时，应用 ID(2 字节)＋ 容器 ID(2 字节)＋ 用户 ID 长度(4 字节)＋用户 ID 值＋输入的待签名数据 P1＝2 时，应用 ID(2 字节)＋ 容器 ID(2 字节)＋ 输入的待签名数据
Le	2	0000	期望设备返回所有签名结果数据

9.6.14.4　命令报文数据域

命令报文数据为应用 ID、容器 ID 和待签名数据。

9.6.14.5　响应报文数据域

本命令响应报文数据为指定容器中 ECC 私钥对输入的待签名数据的签名结果。

相关数据结构定义，

```
typedef struct Struct_ECCSIGNATUREBLOB{
    BYTE r[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE s[ECC_MAX_XCOORDINATE_BITS_LEN/8];
} ECCSIGNATUREBLOB, * PECCSIGNATUREBLOB;
```

表 117　ECCSignData 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	签名密钥对的密钥位长度
4	r	Byte Array	ulBits/8	签名结果的 r 部分
4＋ulBits/8	s	Byte Array	ulBits/8	签名结果的 s 部分

9.6.14.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 118　ECCSignData 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	95	容器中不存对应的密钥对

9.6.15　ECCVerify(ECC 验签)

9.6.15.1　定义与范围

ECCVerify 命令用 ECC 公钥(从外部输入)对数据进行验签。

9.6.15.2　注意事项

任何时候都可以执行此命令。

9.6.15.3　命令报文

表 119　ECCVerify 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	76	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	公钥、签名数据和待验签数据
Le	—	—	无

9.6.15.4　命令报文数据域

表 120　ECCVerify 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	ulBits	ULONG	4	签名密钥的密钥位长度
4	XCoordinate	Byte Array	ulBits/8	ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标

表 120（续）

偏移	数据域	数据类型	字节长度	说　明
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
4＋ulBits/4	ulDataLen	ULONG	4	待验证签名的数据长度
	pbData	Byte Array	ulDataLen	待验证签名的数据
	R	Byte Array	ulBits/8	签名结果的 r 部分
	S	Byte Array	ulBits/8	签名结果的 s 部分

其中待验证签名的数据为待签原始数据的杂凑值。当使用 SM2 算法时，该输入数据为待验签原始数据经过 SM2 签名预处理的结果。

9.6.15.5　响应报文数据域

本命令没有响应报文数据。

9.6.15.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 121　ECCVerify 命令响应状态码

SW1	SW2	意　义
90	00	正确执行，验签成功
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	98	正确执行，但验证签名失败

9.6.16　ECCExportSessionKey（ECC 生成并导出会话密钥）

9.6.16.1　定义与范围

ECCExportSessionKey 命令在设备的指定容器中生成会话密钥并用外部公钥加密导出。

9.6.16.2　注意事项

需要打开应用和容器。

9.6.16.3　命令报文

表 122　ECCExportSessionKey 命令报文编码

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	78	—

表 122（续）

代码	长度 （byte）	值 （Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	外部公钥数据
Le	2	0000	期望设备返回所有结果数据

9.6.16.4　命令报文数据域

表 123　ECCExportSessionKey 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulBits	ULONG	4	外部公钥的密钥位长度
8	XCoordinate	Byte Array	ulBits/8	外部公钥的 XCoordinate，曲线上点的 X 坐标
8＋ulBits/8	YCoordinate	Byte Array	ulBits/8	外部公钥的 YCoordinate，曲线上点的 Y 坐标
8＋ulBits/4	ulAlgId	ULONG	4	会话密钥算法标识

9.6.16.5　响应报文数据域

相关数据结构定义，

```
typedef struct Struct_ECCCIPHERBLOB{
    BYTE XCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8]; //C1 的 X 坐标
    BYTE YCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8]; //C1 的 Y 坐标
    BYTE HASH[32];
    ULONG CipherLen;
    BYTE Cipher[1];
} ECCCIPHERBLOB, *PECCCIPHERBLOB;
```

表 124　ECCExportSessionKey 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	输出密文结构中 C1 的位长度
4	XCoordinate	Byte Array	ulBits/8	输出密文结构中 C1 的 X 坐标
ulBits/8＋4	YCoordinate	Byte Array	ulBits/8	输出密文结构中 C1 的 Y 坐标
ulBits/4＋4	HASH	Byte Array	32	ECCCIPHERBLOB 中的 HASH 数据
ulBits/4＋36	CipherLen	ULONG	4	会话密钥密文数据字节长度

表 124（续）

偏移	数据域	数据类型	字节长度	说　　明
ulBits/4＋40	Cipher	Byte Array	CipherLen	会话密钥密文数据
ulBits/4＋40＋CipherLen	wSessionKeyID	WORD	2	生成的会话密钥 ID

9.6.16.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 125　ECCExportSessionKey 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	99	不支持的会话密钥算法标识

9.6.17　ECCExportSessionKeyEx(ECC 导出会话密钥)

9.6.17.1　定义与范围

ECCExportSessionKeyEx 命令用外部公钥加密导出设备中指定容器中的会话密钥。

9.6.17.2　注意事项

需要打开应用和容器。该函数用于在 RSAExportSessionKey 或 ECCExportSessionKey 函数执行后继续使用其他 ECC 公钥导出会话密钥。

9.6.17.3　命令报文

表 126　ECCExportSessionKeyEx 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	80	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	外部 Ecc 公钥数据
Le	2	0000	期望设备返回所有结果数据

9.6.17.4 命令报文数据域

表 127 ECCExportSessionKeyEx 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	wSessionKeyID	WORD	2	待导出的会话密钥 ID
6	ulBits	ULONG	4	外部公钥的密钥位长度
10	XCoordinate	ByteArray	ulBits/8	外部公钥的 XCoordinate，曲线上点的 X 坐标
10＋ulBits/8	YCoordinate	ByteArray	ulBits/8	外部公钥的 YCoordinate，曲线上点的 Y 坐标

9.6.17.5 响应报文数据域

相关数据结构定义请参考 ECCExportSessionKey 命令的响应报文数据域说明。

表 128 ECCExportSessionKeyEx 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	ulBits	ULONG	4	输出密文结构中 C1 的位长度
4	XCoordinate	Byte Array	ulBits/8	输出密文结构中 C1 的 X 坐标
ulBits/8＋4	YCoordinate	Byte Array	ulBits/8	输出密文结构中 C1 的 Y 坐标
ulBits/4＋4	HASH	Byte Array	32	ECCCIPHERBLOB 中的 HASH 数据
ulBits/4＋36	CipherLen	ULONG	4	会话密钥密文数据字节长度
ulBits/4＋40	Cipher	Byte Array	CipherLen	会话密钥密文数据

9.6.17.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 129 ECCExportSessionKeyEx 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	9F	引用的密钥未找到
6A	99	不支持的会话密钥算法标识

9.6.18 ExtECCEncrypt(ECC 外来公钥加密)

9.6.18.1 定义与范围

ExtECCEncrypt 命令使用外部传入的 ECC 公钥对输入数据做加密运算并输出结果。

9.6.18.2 注意事项

任何时候都可以执行此命令。

9.6.18.3 命令报文

表 130 ExtECCEncrypt 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	7A	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	外部公钥和待加密数据
Le	2	0000	期望设备返回所有加密结果数据

9.6.18.4 命令报文数据域

表 131 ExtECCEncrypt 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	外部公钥的密钥位长度
4	XCoordinate	Byte Array	ulBits/8	外部公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate,曲线上点的 X 坐标
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	外部公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate,曲线上点的 Y 坐标
4＋ulBits/4	ulPlainTextLen	ULONG	4	待加密的数据长度
8＋ulBits/4	pbPlainText	Byte Array	ulPlainTextLen	待加密的数据

9.6.18.5 响应报文数据域

表 132 ExtECCEncrypt 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	ECCCIPHERBLOB 结构中的 ECC 密钥位长度
4	XCoordinate	Byte Array	ulBits/8	ECCCIPHERBLOB 结构中的 XCoordinate,曲线上点的 X 坐标

表 132（续）

偏移	数据域	数据类型	字节长度	说　　明
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	ECCCIPHERBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
4＋ ulBits/4	HASH	Byte Array	32	ECCCIPHERBLOB 中的 HASH 数据
36＋ ulBits/4	CipherLen	ULONG	4	密文数据字节长度
40＋ ulBits/4	Cipher	Byte Array	CipherLen	密文数据

9.6.18.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 133　ExtECCEncrypt 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	9A	非对称加密失败

9.6.19　ExtECCDecrypt（ECC 外来私钥解密）

9.6.19.1　定义与范围

ExtECCDecrypt 命令使用外部传入的 ECC 私钥对输入数据做解密运算并输出结果。

9.6.19.2　注意事项

任何时候都可以执行此命令。

9.6.19.3　命令报文

表 134　ExtECCDecrypt 命令报文编码

代码	长度 （byte）	值 （Hex）	描　　述
CLA	1	80	—
INS	1	7C	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	外部私钥和待解密数据
Le	2	0000	期望设备返回所有解密后明文数据

9.6.19.4 命令报文数据域

表 135 ExtECCDecrypt 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	BitLen	ULONG	4	外部 ECC 密钥对的密钥位长度
4	PrivateKey	Byte Array	BitLen /8	外部私钥 ECCPRIVATEKEYBLOB 结构中的 PrivateKey 数据
4＋BitLen /8	XCoordinate	Byte Array	BitLen /8	ECCCIPHERBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
4＋BitLen /4	YCoordinate	Byte Array	BitLen /8	ECCCIPHERBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
4＋3 * BitLen /8	HASH	Byte Array	32	ECCCIPHERBLOB 中的 HASH 数据，明文的杂凑值
	CipherLen	ULONG	4	密文数据字节长度
	Cipher	Byte Array	CipherLen	密文数据

9.6.19.5 响应报文数据域

本命令没有响应报文数据。

表 136 ExtECCDecrypt 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	ulPlainTextLen	ULONG	4	解密后的明文数据长度
4	pbPlainText	Byte Array	ulPlainTextLen	解密后的明文

9.6.19.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 137 ExtECCDecrypt 命令响应状态码

SW1	SW2	意　义
90	00	正确执行，验签成功
67	00	长度错误（Lc 域给出的长度与数据字节数不一致）
6A	9B	非对称解密失败

9.6.20 ExtECCSign（ECC 外来私钥签名）

9.6.20.1 定义与范围

ExtECCSign 命令使用外部传入的 ECC 私钥对输入数据做签名运算并输出结果。

9.6.20.2 注意事项

任何时候都可以执行此命令。

9.6.20.3 命令报文

表 138 ExtECCSign 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	7E	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	外部私钥和待签名数据
Le	2	0000	期望设备返回所有签名数据

9.6.20.4 命令报文数据域

表 139 ExtECCSign 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	BitLen	ULONG	4	外部 ECC 密钥对的密钥位长度
4	PrivateKey	Byte Array	BitLen /8	外部私钥 ECCPRIVATEKEYBLOB 结构中的 PrivateKey 数据
	ulPlainTextLen	ULONG	4	待签名数据长度
	pbPlainText	Byte Array	ulPlainTextLen	待签名数据

输入的待签名数据为待签原始数据的杂凑值。当使用 SM2 算法时,该待签名数据为待签原始数据经过 SM2 签名预处理的结果,预处理过程遵循 GM/T AAAA。

9.6.20.5 响应报文数据域

表 140 ExtECCSign 响应报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	r	Byte Array	ulBits/8	签名结果的 r 部分
ulBits/8	s	Byte Array	ulBits/8	签名结果的 s 部分

9.6.20.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 141　ExtECCDecrypt 命令响应状态码

SW1	SW2	意　义
90	00	正确执行,验签成功
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	9C	私钥签名失败

9.6.21　GenerateAgreementDataWithECC(ECC 生成密钥协商参数并输出)

9.6.21.1　定义与范围

GenerateAgreementDataWithECC 命令使用 ECC 密钥协商算法,为计算会话密钥而产生协商参数,返回临时 ECC 密钥对的公钥及协商句柄。

9.6.21.2　注意事项

任何时候都可以执行此命令。为协商会话密钥,协商的发起方应首先调用本函数。

9.6.21.3　命令报文

表 142　GenerateAgreementDataWithECC 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	82	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	会话密钥算法标识、发起方的标识
Le	2	0000	期望返回所有数据

9.6.21.4　命令报文数据域

表 143　GenerateAgreementDataWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulAlgId	ULONG	4	会话密钥算法标识
8	ulIDLen	ULONG	4	发起方的 ID 长度
12	pbID	Byte Array	ulIDLen	发起方的 ID

9.6.21.5 响应报文数据域

表 144 GenerateAgreementDataWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	ulBits	ULONG	4	发起方临时 ECC 密钥对的密钥位长度
4	XCoordinate	Byte Array	ulBits/8	发起方临时公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate,曲线上点的 X 坐标
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	发起方临时公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate,曲线上点的 Y 坐标
4＋ulBits/4	hAgreementHandle	ULONG	4	返回的密钥协商句柄

9.6.21.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 145 GenerateAgreementDataWithECC 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6B	02	生成密钥协商数据失败

9.6.22 GenerateAgreementDataAndKeyWithECC(ECC 产生协商数据并计算会话密钥)

9.6.22.1 定义与范围

GenerateAgreementDataAndKeyWithECC 命令使用 ECC 密钥协商算法,产生协商参数并计算会话密钥,输出临时 ECC 密钥对公钥,并返回产生的密钥句柄。

9.6.22.2 注意事项

任何时候都可以执行此命令。本函数由响应方调用。

9.6.22.3 命令报文

表 146 GenerateAgreementDataAndKeyWithECC 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	84	—

表 146（续）

代码	长度（byte）	值（Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	会话密钥算法标识、发起方的 ECC 公钥、发起方临时 ECC 公钥、发起方 ID、响应方 ID
Le	2	0000	期望返回响应方临时 ECC 公钥、对称算法密钥句柄

9.6.22.4　命令报文数据域

表 147　GenerateAgreementDataAndKeyWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	ulAlgId	ULONG	4	会话密钥算法标识
8	ulSponsorBits	ULONG	4	发起方 ECC 密钥对的密钥位长度
	SponsorXCoordinate	Byte Array	ulSponsorBits/8	发起方 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
	SponsorYCoordinate	Byte Array	ulSponsorBits/8	发起方 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
	ulSponsorTempBits	ULONG	4	发起方临时 ECC 密钥对的密钥位长度
	SponsorTempXCoordinate	Byte Array	ulSponsorTempBits/8	发起方临时 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
	SponsorTempYCoordinate	Byte Array	ulSponsorTempBits/8	发起方临时 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
	ulSponsorIDLen	ULONG	4	发起方的 ID 长度
	pbSponsorID	Byte Array	ulSponsorIDLen	发起方的 ID
	ulIDLen	ULONG	4	响应方的 ID 长度
	pbID	Byte Array	ulIDLen	响应方的 ID

9.6.22.5　响应报文数据域

表 148　GenerateAgreementDataAndKeyWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	响应方临时 ECC 密钥对的密钥位长度
4	XCoordinate	Byte Array	ulBits/8	响应方临时公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标

表 148（续）

偏移	数据域	数据类型	字节长度	说　明
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	响应方临时公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
4＋ulBits/4	hSessionKeyID	ULONG	4	返回会话密钥 ID

9.6.22.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 149　GenerateAgreementDataAndKeyWithECC 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6B	01	生成密钥协商数据失败

9.6.23　GenerateKeyWithECC(ECC 计算会话密钥)

9.6.23.1　定义与范围

GenerateKeyWithECC 命令使用 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。

9.6.23.2　注意事项

任何时候都可以执行此命令。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。

9.6.23.3　命令报文

表 150　GenerateKeyWithECC 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	86	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	报文数据长度
DATA	XXXX	XX..XX	密钥协商句柄、响应方 ECC 公钥、响应方临时 ECC 公钥、响应方 ID
Le	2	0004	期望返回协商成功的密钥句柄

9.6.23.4 命令报文数据域

表 151 GenerateKeyWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	hAgreementHandle	ULONG	4	发起方的密钥协商句柄
8	ulResponserBits	ULONG	4	响应方 ECC 密钥对的密钥位长度
	ResponserXCoordinate	Byte Array	ulResponserBits/8	响应方 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
	ResponserYCoordinate	Byte Array	ulResponserBits/8	响应方 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
	ulResponserTempBits	ULONG	4	响应方临时 ECC 密钥对的密钥位长度
	ResponserTempXCoordinate	Byte Array	ResponserTempXCoordinate/8	响应方临时 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
	ResponserTempYCoordinate	Byte Array	ResponserTempXCoordinate/8	响应方临时 ECC 公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
	ulResponserIDLen	ULONG	4	响应方的 ID 长度
	pbResponserID	Byte Array	ulResponserIDLen	响应方的 ID

9.6.23.5 响应报文数据域

表 152 GenerateKeyWithECC 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	hSessionKeyID	ULONG	4	返回会话密钥 ID

9.6.23.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 153 GenerateKeyWithECC 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6B	02	生成协商密钥失败

9.6.24 ExportPublicKey(导出公钥)

9.6.24.1 定义与范围

ExportPubKey 命令用于导出公钥。

9.6.24.2 注意事项

无

9.6.24.3 命令报文

表 154 ExportPubKey 命令报文编码规范

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	88	—
P1	1	00/01	00 表示签名公钥,01 表示加密公钥
P2	1	00	—
Lc	3	00XXXX	数据域长度—
DATA	Lc	XXXXXX	
Le	2	XXXX	期望获取的公钥长度。如果为 0,表示获取实际长度的公钥数据

9.6.24.4 命令报文数据域

命令报文数据域具体见下表。

表 155 ExporPublicKey 命令数据域报文编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	BYTE 数组	2	应用 ID
2	wContainerID	BYTE 数组	2	容器 ID

9.6.24.5 响应报文数据域

响应报文数据域为导出的公钥数据,具体见表。

表 156 ExportPubKey 响应报文编码(RSA 公钥)

偏移	数据域	数据类型	字节长度	说　　明
0	BitLen	ULONG	4	模数的实际位长度
4	Modulus	BYTE 数组	BitLen/8	模数 N
4+BitLen/8	PublicExponent	BYTE 数组	4	公开密钥 E,一般为 0x10001

表 157　ExportPubKey 响应报文编码(ECC 公钥)

偏移	数据域	数据类型	字节长度	说　　明
0	BitLen	ULONG	4	模数的实际位长度
4	XCoordinate	BYTE 数组	BitLen /8	曲线上点的 X 坐标
4+BitLen/8	YCoordinate	BYTE 数组	BitLen /8	曲线上点的 Y 坐标

9.6.24.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 158　ExportPubKey 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	Lc 错误
6A	88	引用的应用不存在
69	82	安全状态不满足
6A	86	参数错误
6A	94	引用的容器不存在
6A	95	对应的密钥对不存在

9.6.25　ImportSessionKey(导入加密会话密钥)

9.6.25.1　定义与范围

ImportSessionKey 命令用于导入密文会话密钥,对会话密钥进行加密操作的公钥为指定应用的指定容器中的加密公钥。

9.6.25.2　注意事项

本命令需要用户权限。

9.6.25.3　命令报文

表 159　ImportSessionKey 命令报文

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	A0	—
P1	1	00	
P2	1	00	

表 159（续）

代码	长度 （byte）	值 （Hex）	描　　述
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	
Le	2	0002	期望获取的会话密钥 ID 长度

9.6.25.4　命令报文数据域

命令报文数据域具体见表.

表 160　ImportSessionKey 命令数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	BYTE 数组	2	应用 ID
2	wContainerID	BYTE 数组	2	容器 ID
4	ulAlgID	ULONG	4	会话密钥算法标识
8	ulWrapedLen	ULONG	4	会话密钥密文长度
12	pbWrapedData	BYTE 数组	ulWrapedLen	会话密钥密文。当容器为 ECC 类型时，密文数据如下表所示进行编码；当容器为 RSA 类型时，此参数为 RSA 公钥加密后的数据

表 161　用 ECC 公钥加密的会话密钥密文

偏移	数据域	数据类型	字节长度	说　　明
0	ulBits	ULONG	4	ECCCIPHERBLOB 结构中的 ECC 密钥位长度
4	XCoordinate	Byte Array	ulBits/8	ECCCIPHERBLOB 结构中的 XCoordinate，曲线上点的 X 坐标
4＋ulBits/8	YCoordinate	Byte Array	ulBits/8	ECCCIPHERBLOB 结构中的 YCoordinate，曲线上点的 Y 坐标
4＋ulBits/4	HASH	Byte Array	32	ECCCIPHERBLOB 中的 HASH 数据
36＋ulBits/4	CipherLen	ULONG	4	密文数据字节长度
40＋ulBits/4	Cipher	Byte Array	CipherLen	密文数据

9.6.25.5　响应报文数据域

响应报文的数据域为对应导入的会话密钥的 ID。

9.6.25.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 162　ImportSessionKey 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	Lc 错误(Lc 不满足解密条件)
6A	88	应用不存在
6A	86	P1/P2 参数错误(算法不支持)
6A	94	引用的容器不存在
6A	95	容器中没有对应的密钥对
69	82	权限不满足
6A	84	空间不足
6A	99	不支持的会话密钥算法标识

9.6.26　ImportSymmKey(导入明文会话密钥)

9.6.26.1　定义与范围

ImportSymmKey 命令用于导入明文形式的会话密钥。

9.6.26.2　注意事项

无

9.6.26.3　命令报文

表 163　ImportSymmKey 命令报文

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	A2	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	
Le	2	00XX	期望获取的会话密钥 ID 长度。

9.6.26.4　命令报文数据域

命令报文数据域具体见表

表 164 ImportSymmKey 命令数据域编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	BYTE 数组	2	应用 ID
2	wContainerID	BYTE 数组	2	容器 ID
4	ulAlgID	ULONG	4	会话密钥算法标识
8	wSymKeyLen	WORD	2	会话密钥长度
10	pbSymKey	BYTE 数组	wSymKeyLen	会话密钥明文

9.6.26.5 响应报文数据域

响应报文的数据域为对应导入的会话密钥的句柄。

9.6.26.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 165 ImportSymmKey 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
6A	88	引用的应用不存在
69	82	安全状态不满足
6A	94	引用的容器不存在
6A	84	空间不足
6A	86	P1/P2 参数错误
6A	99	不支持的会话密钥算法标识

9.6.27 EncryptInit(加密初始化)

9.6.27.1 定义与范围

EncryptInit 命令用于开始加密操作前的参数设置。

9.6.27.2 注意事项

对于指定的密钥，如果存在未结束的加密/解密/散列等操作，EncryptInit 命令将会失败。

9.6.27.3 命令报文

表 166 EncryptInit 命令报文

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	A4	—

表 166（续）

代码	长度 （byte）	值 （Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	
Le	不存在	—	—

9.6.27.4　命令报文数据域

命令报文数据域见表。

表 167　EncryptInit 命令报文编码

偏移	数据域	数据类型	字节长度	说　　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	wKeyID	ULONG	2	密钥 ID
6	AlgID	ULONG	4	算法标识
10	IVLen	WORD	2	起始向量的实际长度
12	IV	BYTE 数组	IVLen	起始向量
12＋ IVLen	PaddingType	ULONG	4	填充方式
16＋ IVLen	FeedBitLen	ULONG	4	反馈值的位长度

9.6.27.5　响应报文数据域

无。

9.6.27.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 168　EncryptInit 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6A	8C	引用的会话密钥不存在
6A	88	引用的应用不存在
67	00	Lc 错误
69	82	权限不满足
6A	94	引用的容器不存在
69	85	使用条件不满足

9.6.28 Encrypt(单组数据加密)

9.6.28.1 定义与范围

Encrypt 命令用于单组数据的加密。

9.6.28.2 注意事项

下传 Encrypt 命令前需要下传 EncryptInit 命令。如果下传 EncryptInit 命令后已经下传过对同一密钥的 Encrypt 命令或 EncryptFinal 命令,则需要重新下传 EncryptInit 命令。

待加密数据长度必须为分组长度的整数倍,设备不对分组数据进行补位填充操作。

9.6.28.3 命令报文

表 169 Encrypt 命令报文

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	A6	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)+容器 ID(2 字节)+密钥 ID(2 字节)+待加密的明文分组数据
Le	2	XXXX	期望回送的密文数据长度。如果为 0,表示回送实际长度的密文数据

9.6.28.4 命令报文数据域

命令报文数据域由应用 ID、容器 ID、密钥 ID(2 字节)及待加密的明文分组数据组成。

9.6.28.5 响应报文数据域

响应报文数据域为期望长度的密文数据。

9.6.28.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 170 Encrypt 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6A	8C	引用的对称密钥不存在
67	00	Lc 错误

表 170（续）

SW1	SW2	意　义
69	85	使用条件不满足
6A	8D	数据错误(密钥句柄与 EncryptInit 命令指定的不符)
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.29　EncryptUpdate(多组数据加密)

9.6.29.1　定义与范围

EncryptUpdate 命令用于多组数据的加密。

9.6.29.2　注意事项

下传 EncryptUpdate 命令前需要下传 EncryptInit 命令。如果下传 EncryptInit 命令后已经下传过对同一密钥的 Encrypt 命令或 EncryptFinal 命令,则需要重新下传 EncryptInit 命令。

待加密数据长度必须为分组长度的整数倍,设备不对分组数据进行补位填充操作。

9.6.29.3　命令报文

表 171　EncryptUpdate 命令报文

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	A8	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋待加密的明文分组数据
Le	2	XXXX	期望回送的密文数据长度。如果为 0,表示回送实际长度的密文数据

9.6.29.4　命令报文数据域

命令报文数据域为待加密的一个或多个明文分组数据。

9.6.29.5　响应报文数据域

响应报文数据域为期望长度的密文数据。

9.6.29.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 172　EncryptUpdate 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
6A	8C	引用的对称密钥不存在
67	00	Lc 错误
69	85	使用条件不满足
6A	8D	数据错误
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.30　EncryptFinal(结束加密)

9.6.30.1　定义与范围

EncryptFinal 命令用于结束一次加密操作,获取剩余加密结果。

9.6.30.2　注意事项

待加密数据长度必须为分组长度的整数倍,设备不对分组数据进行补位填充操作。

9.6.30.3　命令报文

表 173　EncryptFinal 命令报文

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	AA	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)+容器 ID(2 字节)+密钥 ID(2 字节)+最后一段待加密的数据或不存在
Le	2	XXXX	期望回送的密文数据长度。如果为 0,表示回送实际长度的密文数据

9.6.30.4　命令报文数据域

命令报文数据域为待加密的最后一段数据(可以不存在)。

9.6.30.5　响应报文数据域

响应报文数据域为期望长度的密文数据。

9.6.30.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 174 EncryptFinal 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
6A	8C	引用的对称密钥不存在
67	00	Lc 错误
69	85	使用条件不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.31 DecryptInit(解密初始化)

9.6.31.1 定义与范围

DecryptInit 命令用于开始解密操作前的参数设置。

9.6.31.2 注意事项

对于指定的密钥,如果存在未结束的加密/解密/签名/散列等操作,DecryptInit 命令将会失败。

9.6.31.3 命令报文

表 175 DecryptInit 命令报文

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	AC	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	
Le	不存在	—	—

9.6.31.4 命令报文数据域

命令报文数据域的具体内容见表。

表 176 DecryptInit 命令报文编码

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID

表 176（续）

偏移	数据域	数据类型	字节长度	说　明
4	wKeyID	ULONG	2	密钥 ID
10	IVLen	WORD	2	起始向量的实际长度
12	IV	BYTE 数组	IVLen	起始向量
12＋IVLen	PaddingType	ULONG	4	填充方式
16＋IVLen	FeedBitLen	ULONG	4	反馈值的位长度

9.6.31.5　响应报文数据域

无响应报文的数据域。

9.6.31.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 177　DecryptInit 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	Lc 错误
69	82	权限不满足
69	85	使用条件不满足
6A	88	引用的应用不存在
6A	8C	引用的对称密钥不存在
6A	94	引用的容器不存在

9.6.32　Decrypt（单组数据解密）

9.6.32.1　定义与范围

Decrypt 命令用于单组数据的解密。

9.6.32.2　注意事项

下传 Decrypt 命令前需要下传 DecryptInit 命令。如果下传 DecryptInit 命令后已经下传过对同一密钥的 Decrypt 命令或 DecryptFinal 命令，则需要重新下传 DecryptInit 命令。

待解密数据长度必须为分组长度的整数倍，设备不对补位填充的分组数据进行去填充操作。

9.6.32.3　命令报文

表 178　Decrypt 命令报文

代码	长度 （byte）	值 （Hex）	描　述
CLA	1	80	—
INS	1	AE	—

表 178（续）

代码	长度 （byte）	值 （Hex）	描　　述
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋待解密的分组数据
Le	2	XXXX	期望回送的数据长度。如果为 0，表示回送实际长度的数据

9.6.32.4　命令报文数据域

命令报文数据域由应用 ID、容器 ID、密钥 ID(2 字节)和待解密的分组数据组成。

9.6.32.5　响应报文数据域

响应报文数据域为期望长度的解密数据。

9.6.32.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 179　Decrypt 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	Lc 错误
69	85	使用条件不满足
6A	8D	数据错误
6A	88	引用的应用不存在
6A	8C	密钥不存在
6A	94	引用的容器不存在

9.6.33　DecryptUpdate(多组数据解密)

9.6.33.1　定义与范围

DecryptUpdate 命令用于多组数据的解密。

9.6.33.2　注意事项

下传 DecryptUpdate 命令前需要下传 DecryptInit 命令，。如果下传 DecryptInit 命令后已经下传过同一密钥的 Decrypt 命令或 DecryptFinal 命令，则需要重新下传 DecryptInit 命令。

待解密数据长度必须为分组长度的整数倍，设备不对补位填充的分组数据进行去填充操作。

9.6.33.3 命令报文

表 180 DecryptUpdate 命令报文

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	B0	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋待解密的分组数据
Le	2	XXXX	期望回送的数据长度。如果为 0,表示回送实际长度的数据

9.6.33.4 命令报文数据域

命令报文数据域由应用 ID、容器 ID、密钥 ID 和待解密的分组数据组成。

9.6.33.5 响应报文数据域

响应报文数据域为期望长度的解密数据。

9.6.33.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 181 EncryptUpdate 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6A	8D	引用的对称密钥不存在
67	00	Lc 错误
69	85	使用条件不满足
6A	8D	数据错误
6A	88	引用的应用不存在
6A	94	引用的容器不存在

9.6.34 DecryptFinal(结束解密)

9.6.34.1 定义与范围

DecryptFinal 命令用于结束一次解密操作,获取剩余解密结果。

9.6.34.2 注意事项

待解密数据长度必须为分组长度的整数倍,设备不对补位填充的分组数据进行去填充操作。

9.6.34.3 命令报文

表 182 DecryptFinal 命令报文

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	B2	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	数据域长度
DATA	Lc	XXXXXX	应用 ID(2 字节)+容器 ID(2 字节)+密钥 ID(2 字节)+待解密的最后一段数据或不存在
Le	2	XXXX	期望回送的密文数据长度。如果为 0,表示回送实际长度的密文数据

9.6.34.4 命令报文数据域

命令报文数据域由应用 ID、容器 ID、密钥 ID 和待解密的最后一段数据(可以不存在)。

9.6.34.5 响应报文数据域

响应报文数据域为期望长度的解密数据。

9.6.34.6 响应报文状态码

智能密码钥匙可能回送的状态码如下:

表 183 DecryptFinal 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
6A	8D	数据错误
67	00	Lc 错误
69	85	使用条件不满足
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	8C	引用的密钥不存在

9.6.35 DigestInit(密码杂凑初始化)

9.6.35.1 定义与范围

DigestInit 用于初始化密码杂凑计算操作,指定计算密码杂凑的算法。

9.6.35.2 注意事项

任何时候都可以执行此命令。

9.6.35.3 命令报文

表 184 DigestInit 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	B4	—
P1	1	00	—
P2	1	XX	摘要算法标识: 1:SM3; 2:SHA1; 3:SHA256; 其他值暂不定义
Lc	3	00XXXX	当算法标识为 SHA1、SHA256 时,Lc 和 DATA 域不存在; 当算法标识为 SM3 且 Lc 和 DATA 域不存在时,表示使用标准的 SM3 算法进行摘要运算; 当算法标识为 SM3 且 Lc 和 DATA 域存在时,表示执行 SM2 算法签名预处理 1 操作。计算过程遵循 GM/T AAAA。
DATA	XXXX	XX……XX	对于算法标识为 SM3 且 Lc 存在时,输入数据为签名者公钥、签名者 ID 的长度和签名者的 ID 值

9.6.35.4 命令报文数据域

当摘要算法为 SM3 时,命令报文数据由签名者公钥、签名者 ID 的长度和签名者的 ID 值组成。

表 185 DigestInit 命令报文数据域编码

偏移	数据域	数据类型	字节长度	说　　明
0	BitLen	ULONG	4	外部 ECC 密钥对的密钥位长度
4	XCoordinate	Byte Array	BitLen /8	外部公钥 ECCPUBLICKEYBLOB 结构中的 XCoordinate,曲线上点的 X 坐标
	YCoordinate	Byte Array	BitLen /8	外部公钥 ECCPUBLICKEYBLOB 结构中的 YCoordinate,曲线上点的 Y 坐标
	ulIDLen	ULONG	4	签名者 ID 的长度
	pucID	Char Array	ulIDLen	签名者的 ID 值

9.6.35.5 响应报文数据域

无响应数据。

9.6.35.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 186 DigestInit 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	88	引用的应用不存在
6A	94	引用的容器不存在
6A	9D	不支持的摘要算法标识

9.6.36 Digest(单组数据密码杂凑)

9.6.36.1 定义与范围

Digest 用于对单组的消息进行密码杂凑计算。

9.6.36.2 注意事项

调用 Digest 之前,必须调用 DigestInit 初始化密码杂凑计算操作。

9.6.36.3 命令报文

表 187 Digest 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	B6	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	消息数据的长度
DATA	XXXX	XX…XX	消息数据
Le	2	XXXX	密码杂凑结果长度

9.6.36.4 命令报文数据域

命令报文数据由消息数据组成。

9.6.36.5 响应报文数据域

响应报文数据是返回的结果数据。

9.6.36.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 188 Digest 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许

9.6.37 DigestUpdate(多组数据密码杂凑)

9.6.37.1 定义与范围

DigestUpdate 用于对多组的消息进行密码杂凑计算。

9.6.37.2 注意事项

调用 DigestUpdate 之前,必须调用 DigestInit 初始化密码杂凑计算操作。调用 DigestUpdate 以调用 DigestFinal 结束密码杂凑计算操作为结束。

9.6.37.3 命令报文

表 189 Digestupdate 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	B8	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	消息数据的长度
DATA	XX	XX…XX	消息数据

9.6.37.4 命令报文数据域

命令报文数据由消息数据组成。

9.6.37.5 响应报文数据域

无响应数据。

9.6.37.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 190 DigestUpdate 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许

9.6.38 DigestFinal(结束密码杂凑)

9.6.38.1 定义与范围

DigestFinal 用于结束多组消息的密码杂凑计算操作。

9.6.38.2 注意事项

DigestFinal 必须用于 DigestUpdate 之后。

9.6.38.3 命令报文

表 191 DigestFinal 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	BA	—
P1	1	00	—
P2	1	00	—
Lc	3	00XXXX	消息数据的长度
DATA	XX	XX…XX	消息数据
Le	2	XXXX	密码杂凑结果

9.6.38.4 命令报文数据域

命令报文数据为最后一组消息数据。

9.6.38.5 响应报文数据域

响应报文数据是返回的密码杂凑结果。

9.6.38.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 192　DigestFinal 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许

9.6.39　MacInit(消息鉴别码运算初始化)

9.6.39.1　定义与范围

MacInit 用于初始化消息鉴别码计算操作,设置计算消息鉴别码的所需参数。

9.6.39.2　注意事项

任何时候都可以执行此命令。消息鉴别码计算采用分组加密算法的 CBC 模式,将加密结果的最后一块作为计算结果。待计算数据的长度必须是分组加密算法块长的倍数,接口内部不作数据填充。

9.6.39.3　命令报文

表 193　MacInit 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	BC	—
P1	1	00	
P2	1	00	
Lc	3	00XXXX	消息认证计算相关参数的字节数
DATA	XX	XX…XX	消息认证计算相关参数,包括初始向量、初始向量长度、填充方法等

9.6.39.4　命令报文数据域

命令报文数据由下表消息认证计算相关参数组成。

表 194　MacInit 命令报文数据域

偏移	数据域	数据类型	字节长度	说　明
0	wAppID	WORD	2	应用 ID
2	wContainerID	WORD	2	容器 ID
4	wKeyID	ULONG	2	密钥 ID
6	AlgID	ULONG	4	算法标识
10	IVLen	WORD	2	初始向量的实际长度

表 194（续）

偏移	数据域	数据类型	字节长度	说　明
12	IV	BYTE 数组	IVLen	初始向量
12＋ IVLen	PaddingType	ULONG	4	填充方式
16＋ IVLen	FeedBitLen	ULONG	4	反馈值的位长度

9.6.39.5 响应报文数据域

无响应数据。

9.6.39.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 195 MacInit 命令响应状态码

SW1	SW2	意　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
6A	80	在数据字段中的不正确参数
6A	88	应用不存在
6A	94	容器不存在
6A	8C	密钥不存在

9.6.40 Mac(单组数据消息鉴别码运算)

9.6.40.1 定义与范围

Mac 用于计算单一分组数据的消息鉴别码。

9.6.40.2 注意事项

调用 Mac 之前，必须调用 MacInit 初始化消息鉴别码计算操作。

9.6.40.3 命令报文

表 196 Mac 命令报文编码

代码	长度 (byte)	值 (Hex)	描　述
CLA	1	80	—
INS	1	BE	—
P1	1	00	—
P2	1	00	—

表 196（续）

代码	长度 （byte）	值 （Hex）	描　　述
Lc	3	0000XX	命令数据长度
DATA	XX	XX…XX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋消息数据
Le	2	XXXX	Mac 结果的长度

9.6.40.4　命令报文数据域

命令报文数据由应用 ID、容器 ID、密钥 ID 和消息数据组成。

9.6.40.5　响应报文数据域

响应报文数据是返回的 Mac 结果的数据。

9.6.40.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 197　Mac 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	82	安全状态不满足
6A	88	应用不存在
6A	94	容器不存在
6A	8C	密钥不存在

9.6.41　MacUpdate(多组数据消息鉴别码运算)

9.6.41.1　定义与范围

MacUpdate 用于计算多组数据的消息鉴别码。

9.6.41.2　注意事项

调用 MacUpdate 之前，必须调用 MacInit 初始化密码杂凑计算操作。调用 MacUpdate 以调用 MacFinal 结束密码杂凑计算操作为结束。

9.6.41.3　命令报文

表 198　MacUpdate 命令报文编码

代码	长度 （byte）	值 （Hex）	描　　述
CLA	1	80	—

表 198（续）

代码	长度 （byte）	值 （Hex）	描　　述
INS	1	C0	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	命令数据长度
DATA	XX	XX…XX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋消息数据

9.6.41.4　命令报文数据域

命令报文数据由应用 ID、容器 ID、密钥 ID 和消息数据组成。

9.6.41.5　响应报文数据域

无响应数据。

9.6.41.6　响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 199　MacUpdate 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许
6A	88	应用不存在
6A	94	容器不存在
6A	8C	密钥不存在

9.6.42　MacFinal(结束消息鉴别码运算)

9.6.42.1　定义与范围

MacFinal 用于结束多组数据的消息鉴别码计算操作。

9.6.42.2　注意事项

MacFinal 必须用于 MacUpdate 之后。

9.6.42.3 命令报文

表 200 MacFinal 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	C2	—
P1	1	00	—
P2	1	00	—
Lc	3	0000XX	命令数据长度
DATA	XX	XX…XX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)＋消息数据
Le	2	XXXX	返回的消息鉴别码结果长度

9.6.42.4 命令报文数据域

命令报文数据由应用 ID、容器 ID、密钥 ID 和消息数据组成。

9.6.42.5 响应报文数据域

响应报文数据是返回的消息鉴别码结果。

9.6.42.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 201 MacFinal 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许
6A	88	应用不存在
6A	94	容器不存在
6A	88	密钥不存在

9.6.43 DestorySessionKey(销毁会话密钥)

9.6.43.1 定义与范围

DestorySessionKey 命令用于销毁会话密钥。

9.6.43.2 注意事项

任何时候都可以进行此操作。

9.6.43.3 命令报文

表 202 DestorySessionKey 命令报文编码

代码	长度 (byte)	值 (Hex)	描　　述
CLA	1	80	—
INS	1	C4	—
P1	1	00	—
P2	1	00	—
Lc	3	000006	命令数据长度
DATA	6	XX…XX	应用 ID(2 字节)＋容器 ID(2 字节)＋密钥 ID(2 字节)
Le	2	0000	无返回数据

9.6.43.4 命令报文数据域

命令报文数据由应用 ID、容器 ID、密钥 ID 组成。

9.6.43.5 响应报文数据域

无响应报文数据。

9.6.43.6 响应报文状态码

智能密码钥匙可能回送的状态码如下：

表 203 DestorySessionKey 命令响应状态码

SW1	SW2	意　　义
90	00	正确执行
67	00	长度错误(Lc 域给出的长度与数据字节数不一致)
69	86	命令不被允许
6A	88	应用不存在
6A	94	容器不存在
6A	8C	密钥不存在

10 设备协议

10.1 概述

设备应至少支持 USB Mass Storage、HID、CCID 协议中的一种。

10.2 设备识别机制

符合本标准的智能密码钥匙的 VID/PID 需在国家密码管理部门登记注册，并在网站上公开相关信

息,以备接口开发商查询和使用。

10.3 CCID 协议

CCID(Integreted Circuits Card Interface Device)标准规定了一种智能卡接口设备,设备通过 USB 接口与主机或其它嵌入式主机连接,进行符合 CCID 标准的数据通讯,同时设备通过符合 7816 标准协议的接口与智能卡进行通讯。

CCID 协议可参考规范《Specification for Integrated Circuit(s) Cards Interface Devices》,Revision 1.1,2005。

10.4 USB Mass Storage 协议扩展

10.4.1 术语

USB Mass Storage 协议相关术语定义如下表所示。

表 204 USB Mass Storage 协议相关术语

ASC & ASCQ	Additional Sense Code and Additional Sense Code Qualifier(附加状态码和限定的附加状态码) SPC-2 规范中定义的错误代码类型(参见 SPC-2)
BOT	Universal Serial Bus Mass Storage Class Bulk-Only Transport, Rev1.0, 1999(通用串行总线大容量存储类产品猝发传输规范,版本 1.0,1999)
CBWCB	BOT 定义的数据项(详见 BOT)
bCmdCode	CBWCB 域的第 1 个字节
bCmdSubCode	CBWCB 域的第 2 个字节
dCBWDataTransferLength	BOT 定义的数据项(详见 BOT)
dCSWDataResidue	BOT 定义的数据项(详见 BOT)
MSC	Universal Serial Bus Mass Storage Class Specification Overview, Rev1.2, 2003(通用串行总线大容量存储类产品规范概述,版本 1.2,2003)
SCSI	Small Computer Systems Interface(小型计算机系统接口)
SPC-3	Information technology - SCSI Primary Commands-3 (SCSI 基本命令-3)
MMC-4	INFORMATION TECHNOLOGY -Multimedia Commands.4

10.4.2 大容量存储设备(USB Mass Storage)

USB 大容量存储设备类(The USB mass storage device class)是一种计算机和移动设备之间的传输协议,它允许一个通用串行总线(USB)设备来访问主机的计算设备,使两者之间进行文件传输。

当设备(智能 IC 卡读卡器或智能密码钥匙)采用 USB(符合 USB1.1 以上规范)连接端口,符合 USB 大容量存储设备类中 MSC 和 BOT 规范时,插入到安装了 Microsoft Windows 2000/XP/2003/Vista/7/8 和 Linux 2.4.x 以上操作系统的 PC 机或其它终端设备的 USB 口上,PC 机能自动识别智能 IC 卡读卡器或智能密码钥匙,无需安装驱动;只能通过下面定义的接口命令才能对智能 IC 卡读卡器或智能密码钥匙进行操作。

下面将主要描述使用 USB Mass Storage Class - Bulk Only Transport 规范中的 SCSI(Small Com-

puter System Interface，小型计算机系统接口）命令实现 APDU 命令响应对；此设备还需要支持 SCSI 的 REQUEST_SENSE 命令和实现 CDROM 媒体自启动的相关指令，这些指令参见 SPC-3 和 MMC-4 规范；REQUEST_SENSE 命令用于应用管理软件从智能 IC 卡读卡器或智能密码钥匙读取错误类型代码。

10.4.3 APDU 命令响应对

10.4.3.1 发送 APDU 命令（SCSI_OWN_SEND_APDU）

10.4.3.1.1 发送 APDU 扩展指令定义

通过 USB Mass Storage 协议给智能密码钥匙发送 APDU 指令的格式定义如下表 205。

表 205 USB Mass Storage 协议发送 APDU 扩展指令定义

Offset	Field	Size	Value	Description
0	bcmdCode	1	0xFE	
1	bCmdSubCode	1	0x01	
2	byMagicCode	9	“GMCAPIDFS”的 ASCII 码表示	

10.4.3.1.2 发送 APDU 扩展数据格式定义

通过 USB Mass Storage 协议给智能密码钥匙发送 APDU 命令的扩展数据格式如下表 206 定义。

表 206 USB Mass Storage 协议扩展数据格式定义

Offset	Field	Size	Value	Description
0	Tag	1	NAD	NAD(node address) 0x11——为智能 IC 卡读卡器命令，如果读写器不支持此命令，则将命令发给主卡； 0x12——为操作智能密码钥匙 其它——保留，暂为无效指令
1	Length	2	LEN	采用高位在前(Big Endian)格式，双字节表示后续 APDU 命令的长度
3	APDU 命令头	4	CLA	具体描述参见第 7 部分“命令头、数据字段和响应尾标用的编码约定”
			INS	
			P1	
			P2	
7	APDU 命令体	LEN-4	Lc	
			Data	
			Le	

表 206 传输的数据总长度最大为 65536 个字节。

10.4.3.2 读取 APDU 响应(SCSI_OWN_READ_ADPU_RET)

10.4.3.2.1 读取 APDU 响应指令定义

通过 USB Mass Storage 协议给智能密码钥匙发送相关取响应指令的格式定义如下表 207。

表 207 USB Mass Storage 协议取 APDU 响应扩展指令定义

Offset	Field	Size	Value	Description
0	bcmdCode	1	0xFE	
1	bCmdSubCode	1	0x02	

10.4.3.2.2 读取 APDU 响应数据格式定义

通过 USB Mass Storage 协议读取 APDU 命令响应的数据格式如下表 208 定义。

表 208 USB Mass Storage 协议取 APDU 响应的数据格式定义

Offset	Field	Size	Value	Description
0	Tag	1	NAD	NAD(node address) 0x11——为智能 IC 卡读卡器命令,如果读写器不支持此命令,则将命令发给主卡; 0x12——为操作智能密码钥匙 其它——保留,暂为无效指令
1	Length	2	LEN	采用高位在前(Big Endian)格式,双字节表示后续 APDU 命令的长度
3	APDU 命令响应数据+状态字	LEN	Data	具体描述参见第 7 部分“命令头、数据字段和响应尾标用的编码约定”
			SW1SW2	

以上命令执行的等待时间默认最长为 60 s。

10.4.3.2.3 USB Mass Storage 设备相关 SCSI 命令集(CDROM 媒体自启动)

表 209 USB Mass Storage 设备相关 SCSI 命令集

SCSI Command	Instruction	Description
0x00	TEST UNIT READY	
0x03	REQUEST SENSE	
0x12	INQUIRY	
0x46	GET CONFIGURATION	
0x1B	START STOP UNIT	
0x23	READ FORMAT CAPACITIES	
0x1E	PREVENT ALLOW MEDIUM REMOVAL	
0x25	READ CAPACITY	
0x28	READ(10)	

参考规范：

1. INFORMATION TECHNOLOGY-Multimedia Commands. 4（MMC-4）
2. Information technology-SCSI Block Commands-2（SBC-2）
3. Information technology-SCSI Primary Commands-3（SPC-3）

10.4.4 错误代码类型

能IC卡读卡器或智能密码钥匙必须支持的Sense Key和ASC & ASCQ如下表210和表211定义。

表210 Sense Key定义

Sense Key	Description
0	没有错误
1	异常

表211 ASC & ASCQ定义

ASC & ASCQ	Description
0x2000	非法的命令代码
0xCACB	防操作冲突 为了避免设备受多线程和进程的操作冲突，底层定义了错误代码：0xCACB； 底层处理应用接口命令操作的过程为： 1. 收到APDU操作命令，执行命令 2. 等待PC发来“接收响应”的CBW，如果接收到PC发来“接收响应”的CBW，返回响应，继续1；如果接收到PC发来“发送命令”的CBW，返回0xCACB的错误代码，继续2 底层在等待接收PC发来“接收响应”的CBW过程中，设置最多等待CBW命令包个数，如果在等待时间内仍未收到“接收响应”的CBW，将停止等待
0xCACD	增加等待时间扩展功能： 1. 一般操作设备的APDU命令等待时间设为1分钟 2. 当“发送命令”接收到0xCACD的SenseCode时，PC继续发送“接收响应”，并将该“接收响应”的命令等待时间设为3分钟。如果“接收响应”命令接收到0xCACD的SenseCode，则继续发送“接收响应”，该“接收响应”的命令等待时间设为3分钟。如此循环10次。如果“接收响应”命令还是接收到0xCACD的SenseCode，则退出

10.5 HID协议扩展

10.5.1 术语

HID协议相关术语定义如下表212所示。

表212 HID协议相关术语

HID	Human Interface Devices
Set report	主机通过控制端点向HID设备发送数据请求
Get report	主机通过控制端点从HID设备接收数据请求

10.5.2 HID 协议简介

HID(Human Interface Device)人机接口设备是计算机直接与人交互的设备。当设备(智能 IC 卡读卡器或智能密码钥匙)采用 USB(符合 USB1.1 以上规范)协议,且符合 USB HID 规范时,插入到安装了 Microsoft Windows 2000/XP/2003/Vista/7/8、Linux 和 Mac OS 操作系统的 PC 机或其它终端设备的 USB 口上,PC 机能自动识别智能 IC 卡读卡器或智能密码钥匙,无需安装驱动。

10.5.3 数据包格式

10.5.3.1 控制传输

10.5.3.1.1 概述

HID 设备通过端点 EP0 采用控制传输,实现数据(APDU)的传输。

10.5.3.1.2 数据接收

通过 EP0 接收数据时,在获取到一个数据包时需要首先分析收到的数据,看其是否是上位机发送的级联数据。如果是,则继续接收上位机命令,直到级联数据接收完毕为止。

级联接收的具体流程如下:

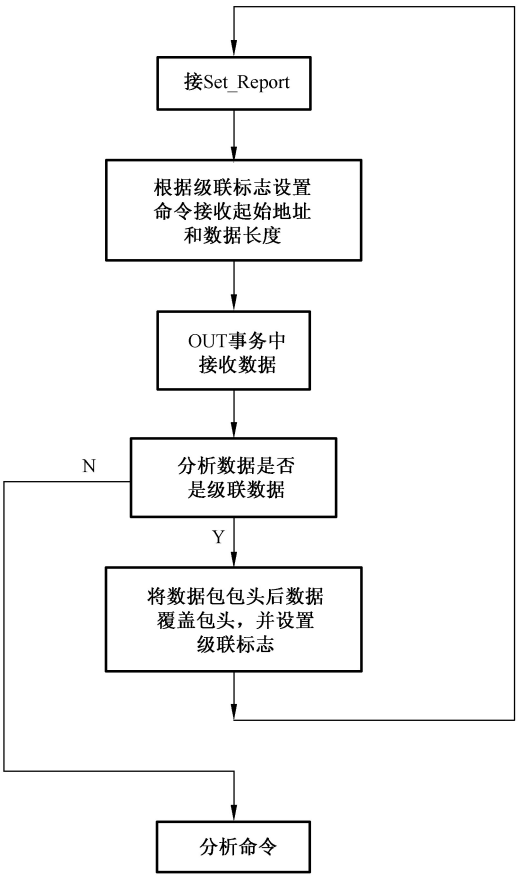


图 8 HID 设备数据接收流程

上位机发送的级联数据每包 64 字节,包头 3 字节,最后一个包实际数据除去包头可能不到 61 字节,但是还是将不到 61 字节的地方补 0。所以接收的时候始终按 64 字节长度字节去接收。

接收到的级联包实际是包含了包头的,给 COS 的命令是要去掉这 3 字节包头。然后按下图搬移数据:

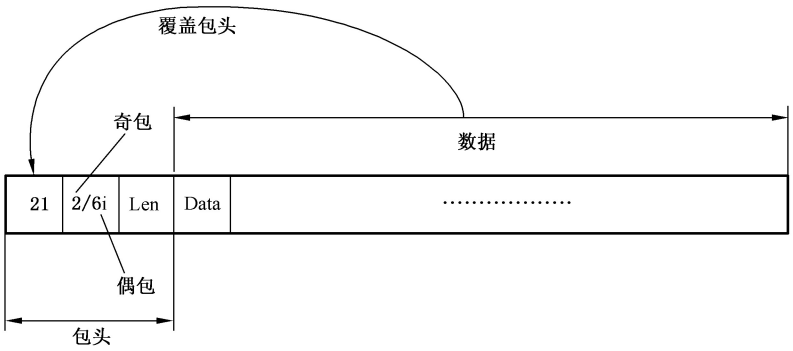


图 9　HID 设备数据包组装示意图

注：最后一包第 2 字节为 0i。

但是这样的操作方式特别浪费时间，实际在操作的时候接收的前 3 字节去掉，从第 4 字节才写入到对应数据区。这样就省去了数据的搬移。在接收数据的时候第一个 64 字节的包全部接收，接收之后分析前 3 字节，根据情况判断级联，发现级联之后的包前 3 字节不写入数据区，从第 4 字节开始写入数据区，但需要对第 2 字节进行分析是否之后还有级联包。

10.5.3.1.3　数据发送

通过 EP0 发送数据时，在发送数据前首先分析数据长度是否需要级联发送。

级联发送的具体流程如下：

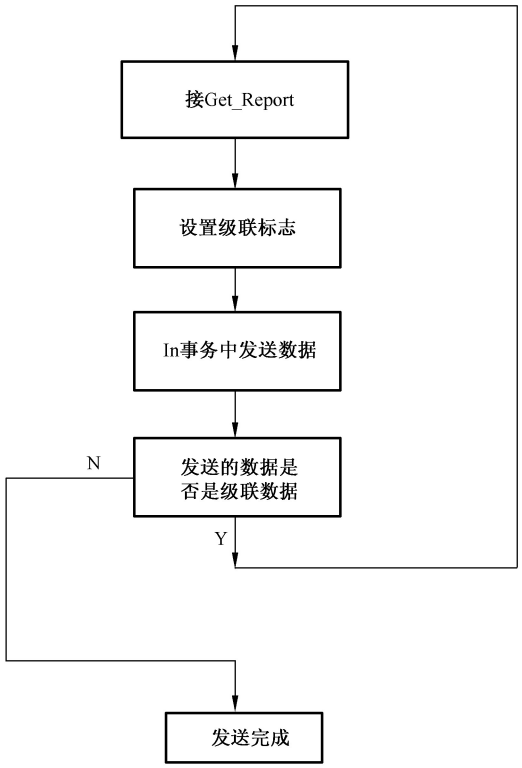


图 10　HID 设备数据发送流程

发送的方法是将待发送的数据按 61 字节拆开，然后在每个包加上包头，最后一个包可能不到 61 字节，也只要加上包头就可以了，包头如下：

级联非最后一包包头:0x21 0x2i 0x3d,i 是第几个级联包。

级联最后一包包头:0x21 0x0i Len,Len 是最后一包出去包头的字节长度,这个长度包含了最后 2 字节的状态字。

10.5.3.1.4 HID 延时处理

HID 设备在处理上位机的命令数据时,可能需要一定的时间,这段时间可能超出 HID 的可接受的响应时间范围。这个时候就需要延时,在 COS 处理命令之间,回复一个响应数据包:0x21、0xC3、0x00,通知上位机延时,在延时包发送后,最大可以延时 5 秒,也就是 5 秒之内 COS 还没有处理好数据,则还需要回复延时包。

延时的处理实际上是交给了定时器去处理,当需要调用 COS 时就开启定时器和溢出中断使能,关掉串行中断使能。

当定时器溢出时,关定时器,设置回复延时包,开串行中断使能。

当获取到 Get_Report 包时,关串行中断,开启定时器。

延时包的处理流程如下:

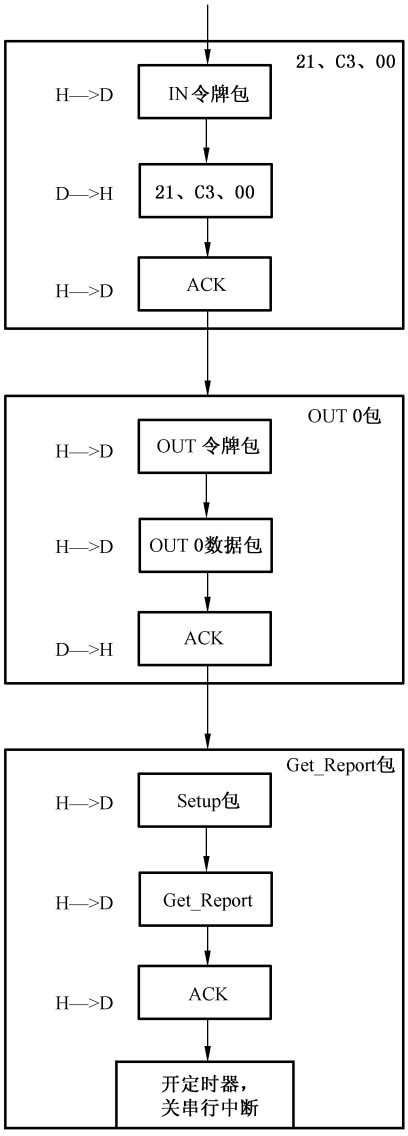


图 11 HID 设备数据延时处理流程

对延时包的处理要将 21、C3、00，OUT 0 包，Get_Report 作为一个整体来完成。这个整体的完成而且是在定时器溢出程序中完成的，在里面采用查询的方式。

附 录 A
（规范性附录）
设备返回码定义和说明

SW	说 明
9000	正确执行
63CX	认证失败，还剩下 X 次重试机会
6700	长度错误(Lc 域给出的长度与 DATA 域数据长度不一致)
6982	安全状态不满足
6983	认证方法被锁定
6984	引用的数据无效
6985	使用的条件不满足
6986	命令不被允许
6988	SM 数据对象不正确
698A	没有选择(打开)应用
6A80	在数据字段中的不正确参数
6A81	功能不被支持
6A84	无足够的文件存储空间
6A86	不正确的参数 P1-P2
6A88	引用的数据未找到
6A89	应用已经存在
6A8A	指定的应用已打开
6A8B	指定的应用不存在
6A8C	引用的对称密钥不存在
6A8D	数据错误
6A90	已有打开的应用，当前设备不支持同时打开多个应用
6A91	指定的容器不存在
6A92	文件已经存在
6A93	指定的文件不存在
6A94	引用的容器未找到
6A95	容器中没有对应的密钥对
6A96	指定类型的证书不存在
6A97	数据写入失败
6A98	验证签名失败
6A99	不支持的会话密钥算法标识
6A9A	非对称加密失败

SW	说　明
6A9B	非对称解密失败
6A9C	私钥签名失败
6A9D	不支持的摘要算法标识
6A9E	还有更多数据需要上传,接口层需重新发送指令获取后续数据
6A9F	引用的密钥未找到
6B00	给定的偏移值超出文件长度
6B01	生成密钥协商数据失败
6B02	生成协商密钥失败
6CXX	错误的长度 Le,SW2 指示准确的长度
6E00	CLA 错误,指定的类别不被支持
6E01	待返回数据长度超出最大缓冲区长度
6E02	指定的容器已存在

附 录 B
（规范性附录）
安全报文计算说明

假定分组长度为 L，待加密数据长度为 Ld（最大长度为 65 535）。

数据加密运算

第一步：在待加密数据前填充两字节的数据长度（以 LittleEndian 方式表示），作为加密过程的输入数据；

第二步：将经第一步处理后的数据块分成 L 字节为单位的数据块，标识为 D1、D2……Dn 等；

第三步：如果最后一组数据块长度为 L 字节，必须在后加上 1 字节 0x80 及 L-1 字节 0x00，转到第四步；如果最后一组数据块长度不足 L 字节，则在后加上 0x80，如果长度达到 L 字节，转到第四步；否则在其后再加上 0x00，直到长度达到 L 字节。

第四步：对上述每一个数据块使用相应的密钥进行加密运算。

第五步：计算结束后 ，将所有运算后的数据块依照原有顺序连接到一起，即为加密运算后结果。

数据解密运算

第一步：对待解密数据按 L 字节进行分组；

第二步：对每一个分组使用相应的密码进行解密运算，所有解密后的数据按照原有序列连接到一起；

第三步：去掉前面填充的数据长度和最后的填充数据，得到原始明文。

MAC 计算

MAC 是使用命令的所有元素（包括命令头）产生的，MAC 是命令数据域中一部分，长度为 4 字节。

MAC 计算步骤如下：

第一步：应用向设备发送取随机数指令，从设备取回 8 字节随机数。

第二步：将设备回送的 8 字节随机数加上 L-8 个字节 0x00，得到 L 字节结果作为初始值。

第三步：按照顺序将以下数据连接成数据块：

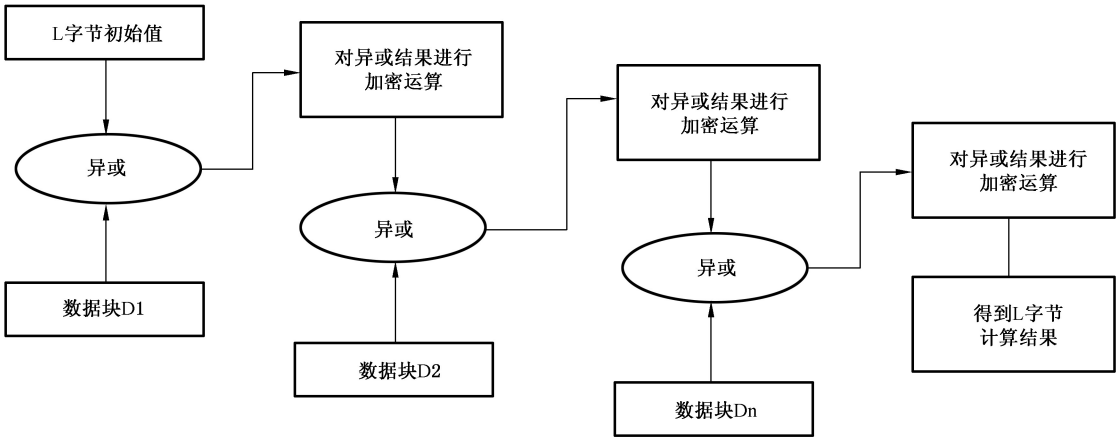
CLA、INS、P1、P2、Lc＋4、Data；

必须将 CLA 的低半字节置为 0x4。

第四步：将数据块分成 L 字节为单位的数据块，标识为 D1、D2……Dn 等。

第五步：如果最后一组数据块长度为 L 字节，必须在后加上 1 字节 0x80 及 L-1 字节 0x00，转到第六步；如果最后一组数据块长度不足 L 字节，则在后加上 0x80，如果长度达到 L 字节，转到第六步；否则在其后再加上 0x00，直到长度达到 L 字节。

第六步：对上述数据块使用相应的密钥进行加密，如下图：



第七步:将上述加密得到的结果从左侧取 4 字节长度数据为 MAC。

附 录 C
（资料性附录）
编 程 范 例

本附录对规范中常用指令提供了编程范例，供应用开发商参考，应用开发商可以根据实际情况进行相应的修改后使用。本附录中的编程范例使用的开发语言为 C 语言，范例中使用的函数、数据类型和常量定义见 GM/T BBBB

1. 设备认证

```
BYTE *CLA, *INS, *P1, *P2, *Lc, *Data, *Le;
BYTE SW1,SW2;
BYTE CommAPDU[64],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;
HANDLE hDev;

BYTE Rand[16];
BYTE AuthData[16];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
memset(CommAPDU,0,64);
memset(RespAPDU,0,64);

./*打开设备,并取得设备句柄 hDev*/
.
.
.
/*取随机数*/
*CLA=0x80;
*INS=0x50;
*P1=0x00;
*P2=0x00;
Le=P2+1;
*Le=0x00;
Le++;
*Le=0x00;
Le++;
*Le=0x08;
CommLen=7;
```

```
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
memcpy(Rand,RespAPDU,RespLen-2);

/*得到加密认证数据*/
/*……
```

使用外部设备或软件，使用设备认证密钥，参考 GM/T BBBB 8.2.3，加密随机数得到认证数据，并放在 AuthData 数组中。在本例中，加密采用 SM1 算法，加密后的设备认证数据长度为 16 字节。

```
…… */
/*设备认证*/
 *CLA=0x80;
 *INS=0x10;
 *P1=0x00;
 *P2=0x00;/*采用默认的 SM1 算法*/
Lc=P2+1;
 *Lc=0x00;
Lc++;
 *Lc=0x00;
Lc++;
 *Lc=0x10;
Data=Lc+1;
memcpy(Data,AuthData,16);
CommLen = 23;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
```

```
{
  /* 错误处理 */
  ……
}
```

2. 修改设备认证密钥

```
BYTE *CLA, *INS, *P1, *P2, *Lc, *Data, *Le;
BYTE SW1,SW2;
BYTE CommAPDU[64],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;

BYTE NewAuthKey[16];
BYTE EncryptedNewAuthKey[16];
BYTE Mac[8];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
memset(CommAPDU,0,64);
memset(RespAPDU,0,64);

.
.
.
/* 得到加密认证数据 */
/* ……
```

i. 输入新的设备认证密钥并保存在 NewAuthKey 数组中；

ii. 使用外部设备或软件，使用原设备认证密钥，参照《智能密码钥匙密码应用接口数据格式规范》附录 B，加密 NewAuthKey 并保存在 EncryptedNewAuthKey 数组中，；

iii. 使用外部设备或软件，使用原设备认证密钥参照《智能密码钥匙密码应用接口数据格式规范》附录 B，对 NewAuthKey 生成认证码并保存在 Mac 数组中；

```
…… */
/* 修改设备认证密钥 */
 *CLA=0x84;
 *INS=0x12;
 *P1=0x00;
 *P2=0x00;
Lc=P2+1;
 *Lc=0x00;
Lc++;
 *Lc=0x00;
```

```
Lc++;
*Lc=0x14;
Data=Lc+1;
memcpy(Data, EncryptedNewAuthKey,16);
memcpy(Data+16,Mac,4);
CommLen = 27;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
```

3. 设置设备标签

```
  BYTE NewLabel[32]="newlabel";
  ULONG LabelLen=8;
  *CLA=0x80;
  *INS=0x02;
  *P1=0x00;
  *P2=0x00;
  Lc=P2+1;
  *Lc=0x00;
  Lc++;
  *Lc=0x00;
  Lc++;
  *Lc=(BYTE)LabelLen;
  Data=Lc+1;
  memcpy(Data,NewLabel,NewLabelLen);
  CommLen=7+NewLabelLen;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
```

```
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
```

4. 添加应用

```
/* 采用设备认证接口,完成设备认证 */
/*
…
*/
/**设置应用信息*/

DWORD BigEndianDWORD(DWORD m)
{
  //将输入四字节整数转换为 BigEndian 编码
}

WORD BigEndianWORD(WORD m)
{
  //将输入两字节整数转换为 BigEndian 编码
}

APPLICATIONINFO appInfo;
CHAR szApplicatinName[32] = "appName";
CHAR  szAdminPin[16] = "adminPin";
CHAR  szUserPin[16] = "userPin";

memset(&appInfo,0,sizeof(appInfo));
strcpy(appInfo. szApplicatinName, szApplicatinName);
strcpy(appInfo. szAdminPin, szAdminPin);
strcpy(appInfo. szUserPin, szUserPin);
appInfo. bApplicationNameLen = BigEndianDWORD(strlen(szApplicatinName));
appInfo. bAdminPinLen= BigEndianDWORD(strlen(szAdminPin));
appInfo. bUserPinLen = BigEndianDWORD(strlen(szUserPin));
appInfo. dwAdminPinRetryCount = 10;
appInfo. dwUserPinRetryCount= 10;
appInfo. dwCreateFileRights = 3;
appInfo. byContainerNum = 16;
appInfo. byCertNum = 16;
appInfo. wFileNum = BigEndianWORD(256);

/*添加应用指令数据 */
```

```
BYTE *CLA, *INS, *P1, *P2, *Lc, *Data;
BYTE SW1,SW2;
BYTE CommAPDU[64+sizeof(appInfo)],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;

BYTE Rand[16];
BYTE AuthData[16];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;

memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));



*CLA=0x80;
*INS=0x20;  // CreateApplication
*P1=0x00;
*P2=0x00;
Lc[2] = sizeof(appInfo);
memcpy(Data,&appInfo,sizeof(appInfo));
CommLen=7+sizeof(appInfo);

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
```

5. 删除应用

```
/* 采用设备认证接口,完成设备认证 */
/*
...
*/
/* *设置应用名称信息*/
CHAR szApplicatinName[32] = "appName";


/* 添加应用指令数据 */
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data;
BYTE SW1,SW2;
BYTE CommAPDU[64+sizeof(appInfo)],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;

BYTE Rand[16];
BYTE AuthData[16];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;

/* 删除应用 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));

*CLA=0x80;
*INS=0x24;  // DeleteApplication
*P1=0x00;
*P2=0x00;
Lc[2]=strlen(szApplicatinName);
memcpy(Data, szApplicatinName, strlen(szApplicatinName));

CommLen=7+ strlen(szApplicatinName);

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
```

```
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /* 错误处理 */
  ……
}
```

6. **修改 PIN**

```
/* PIN 码信息 */
CHAR szOldAdminPin[16] = "oldAdminPin";
CHAR szNewAdminPin[16] = "newAdminPin";

/* 指令数据 */
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
BYTE SW1,SW2;
BYTE CommAPDU[64+sizeof(appInfo)],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;

BYTE Rand[16];
BYTE AuthData[16];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;

/* 打开应用 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));

*CLA=0x80;
*INS=0x26;  // OpenApplication
*P1=0x00;
*P2=0x00;
Lc[2]=(BYTE)strlen(szApplicatinName);
```

```
memcpy(Data, szApplicatinName, strlen(szApplicatinName));

Le = Data + strlen(szApplicatinName);
Le[1] = 10;

CommLen=9+ strlen(szApplicatinName);

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}

if(RespLen ! = 12 )  // 返回数据格式错误,无法获取 wAppId
{
  /*错误处理*/
  ……
}

/* 获取 AppID*/
BYTE pAppId[2];
memcpy(pAppId, RespAPDU + 8,2);


/* 取随机数 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x80;
*INS=0x50;  // GenRandom
*P1=0x00;
*P2=0x00;
Le=P2+1;
Le[2]=0x08;
CommLen=7;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
```

```
if (RetValue ! = SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /* 错误处理 */
  ……
}

if(RespLen ! = 10 )  // 返回数据格式错误,随机数长度不正确
{
  /* 错误处理 */
  ……
}
BYTE pRandom[8];
memcpy(pRandom, RespAPDU,8);

/* 对原 PIN 码进行 SHA1 运算 */
CHAR sha1Hash[20];
SHA1(szOldAdminPin,16,sha1Hash);

/* 加密 NewAdminPIN */
BYTE encNewAdminPIN [16];
/* SM4 (…)函数由软件实现 */
SM4(sha1Hash,16, szNewAdminPin,16, encNewAdminPIN,16); //

/* 修改 PIN 码 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x84;
*INS=0x16;  // ChangePIN
*P1=0x00;
*P2=0x00;  // 管理员 PIN 码
Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=20;
```

```
    Data = Lc++;
    memcpy(Data, pAppId,2);    // pAppId
    memcpy(Data+2, encNewAdminPIN,16); // encNewAdminPIN

    /* 计算 Mac，SM4_MAC 由软件实现 */
    SM4_MAC(sha1Hash,16, pRandom, 8, CommAPDU,7+2+16, CommAPDU+7+2+16,4);

  /*/ 发送指令 */
  CommLen=7+22;
  RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
  if (RetValue != SAR_OK)
  {
    /*错误处理*/
    ……
  }
  SW1=RespAPDU[RespLen-2];
  SW2=RespAPDU[RespLen-1];
  if ((SW1 != 0x90)||(SW2 != 0x00))
  {
    /*错误处理*/
    ……
  }
```

7. 校验 PIN

```
  /*PIN 码信息*/
  CHAR szAdminPin[16] = "adminPin";



  /* 指令数据 */
  BYTE *CLA, *INS, *P1, *P2, *Lc, *Data, *Le;
  BYTE SW1,SW2;
  BYTE CommAPDU[64+sizeof(appInfo)],RespAPDU[64];
  ULONG CommLen;
  ULONG RespLen;
  ULONG RetValue;

  BYTE Rand[16];
  BYTE AuthData[16];

  CLA=CommAPDU;
  INS=CommAPDU+1;
  P1=CommAPDU+2;
  P2=CommAPDU+3;
  Lc= CommAPDU+4;
```

```
Data= CommAPDU+7;

/* 打开应用 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));

*CLA=0x80;
*INS=0x26;  // OpenApplication
*P1=0x00;
*P2=0x00;
Lc[2]=(BYTE)strlen(szApplicatinName);
memcpy(Data, szApplicatinName, strlen(szApplicatinName));

Le = Data + strlen(szApplicatinName);
Le[1] = 10;

CommLen=9+ strlen(szApplicatinName);

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
/* 返回数据格式错误,无法获取 wAppId */
if(RespLen ! = 12 )
{
  /*错误处理*/
  ……
}

/* 获取 AppID*/
BYTE pAppId[2];
memcpy(pAppId, RespAPDU + 8,2);

  /** 取随机数 */
```

```
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x80;
*INS=0x50;  // GenRandom
*P1=0x00;
*P2=0x00;
Le=P2+1;
Le[2]=0x08;
CommLen=7;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
/* 返回数据格式错误,随机数长度不正确 */
if(RespLen ! = 10 )
{
  /*错误处理*/
  ……
}
BYTE pRandom[8];
memcpy(pRandom, RespAPDU,8);

/** 对 PIN 码进行 SHA1 运算*/
CHAR sha1Hash[20];
/* 进行 SHA1 运算,SHA1(…)函数由软件实现 */
SHA1(szAdminPin,16,sha1Hash);

/**加密 pRandom*/
BYTE encRandom [16];
/* SM4 (…)函数由软件实现 */
SM4(sha1Hash,16, pRandom,8, encRandom,16);

/**校验 PIN 码*/
memset(CommAPDU,0,sizeof(CommAPDU));
```

```
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x80;
*INS=0x18;  // VerifyPIN
*P1=0x00;
*P2=0x00;  // 管理员 PIN 码
Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=18;
Data = Lc++;
memcpy(Data, pAppId,2);   // pAppId
memcpy(Data+2, encRandom,16); // encRandom


CommLen=7+18;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
```

8. PIN 解锁

```
/*PIN 码信息*/
CHAR szAdminPin[16] = "adminPin";
CHAR szNewUserPin[16] = "newUserPin";

/* 指令数据 */
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
BYTE SW1,SW2;
BYTE CommAPDU[64+sizeof(appInfo)],RespAPDU[64];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;
```

```
BYTE Rand[16];
BYTE AuthData[16];

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;

/* 打开应用 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));

*CLA=0x80;
*INS=0x26;  // OpenApplication
*P1=0x00;
*P2=0x00;
Lc[2]=strlen(szApplicatinName);
memcpy(Data, szApplicatinName, strlen(szApplicatinName));

Le = Data + strlen(szApplicatinName);
Le[1] = 10;

CommLen=9+ strlen(szApplicatinName);

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}

if(RespLen != 12 )  // 返回数据格式错误,无法获取 wAppId
{
```

```
  /* 错误处理 */
  ……
}

/* 获取 AppID */
BYTE pAppId[2];
memcpy(pAppId, RespAPDU + 8,2);

/* 取随机数 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x80;
*INS=0x50;  // GenRandom
*P1=0x00;
*P2=0x00;
Le=P2+1;
Le[2]=0x08;
CommLen=7;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /* 错误处理 */
  ……
}

if(RespLen ! = 10 )  // 返回数据格式错误,随机数长度不正确
{
  /* 错误处理 */
  ……
}
BYTE pRandom[8];
memcpy(pRandom, RespAPDU,8);

/* 对原 PIN 码进行 SHA1 运算 */
CHAR sha1Hash[20];
/* 进行 SHA1 运算,SHA1(…)函数由软件实现 */
```

```
SHA1(szAdminPin,16,sha1Hash);

/* 加密 NewUserPin */
BYTE encNewUserPIN [16];
/* SM4 (…)函数由软件实现 */
SM4(sha1Hash,16, szNewUserPin,16, encNewUserPIN,16);

/* 解锁 PIN 码 */
memset(CommAPDU,0,sizeof(CommAPDU));
memset(RespAPDU,0, sizeof(RespAPDU));
*CLA=0x84;
*INS=0x1A;  // UnlockPIN
*P1=0x00;
*P2=0x00;  // 管理员 PIN 码
Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=18;
Data = Lc++;
memcpy(Data, pAppId,2);   // pAppId
memcpy(Data+2, encNewUserPIN,16); // encNewAdminPIN

/* 计算 Mac , SM4_MAC 由软件实现 */
SM4_MAC(sha1Hash,16, pRandom, 8, CommAPDU,7+2+16, CommAPDU+7+2+16,4);

CommLen=7+22;
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /* 错误处理 */
  ……
}
```

9. 创建密钥容器

/*以下编程范例演示打开已存在的名称为"CAAPP"应用,并创建名称为"12345678"的密钥容

```
器*/
    BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
    BYTE SW1,SW2;
    BYTE CommAPDU[64],RespAPDU[64];
    ULONG CommLen;
    ULONG RespLen;
    ULONG RetValue;
    WORD *pwAppID;
    WORD wAppID;


    CLA=CommAPDU;
    INS=CommAPDU+1;
    P1=CommAPDU+2;
    P2=CommAPDU+3;
    memset(CommAPDU,0,64);
    memset(RespAPDU,0,64);

    /*打开名称为"CAAPP"应用*/
    *CLA=0x80;
    *INS=0x26;
    *P1=0x00;
    *P2=0x00;

    Lc=P2+1;
    *Lc=0x00;
    Lc++;
    *Lc=0x00;
    Lc++;
    *Lc=0x05;


    Data=Lc+1;
    memcpy(Data," CAAPP",5);


    RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
    if (RetValue != SAR_OK)
    {
      /*错误处理*/
      ……
    }
    SW1=RespAPDU[RespLen-2];
```

```
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}

pwAppID= RespAPDU+8;
wAppID = *pwAppID;
.
/*创建密钥容器,容器名称为"12345678"*/
*CLA=0x80;
*INS=0x40;
*P1=0x00;
*P2=0x00;

Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=0x0A;


Data=Lc+1;
pwAppID = Data;
*pwAppID = BigEndianWORD(wAppID);
Data+=2;
memcpy(Data,"12345678",8);

Le= Data +8;
*Le=0x00;
Le++;
*Le=0x02;
CommLen = 9+8;

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
```

```
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}

/*关闭名称为"CAAPP"应用*/
*CLA=0x80;
*INS=0x28;
*P1=0x00;
*P2=0x00;

Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=0x02;

Data=Lc+1;
pwAppID= Data
*pwAppID = BigEndianWORD(wAppID);

RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
```

10. 删除密钥容器

```
/*以下编程范例演示打开已存在的名称为"CAAPP"应用,并删除名称为"12345678"的密钥容器*/
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
BYTE SW1,SW2;
BYTE CommAPDU[64],RespAPDU[64];
```

```
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;
WORD *pwAppID;
WORD wAppID;

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
memset(CommAPDU,0,64);
memset(RespAPDU,0,64);

/*打开名称为"CAAPP"应用*/
*CLA=0x80;
*INS=0x26;
*P1=0x00;
*P2=0x00;

Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=0x05;
Data=Lc+1;
memcpy(Data," CAAPP",5);

RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}

pwAppID= RespAPDU+8;
```

```
wAppID = *pwAppID; //此处需调整为机器字节序


/*此处请进行 VerifyPin 用户 PIN 码认证 */
……


.
/*删除已存在的容器名称为"12345678" 密钥容器*/
*CLA=0x80;
*INS=0x48;
*P1=0x00;
*P2=0x00;
Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=0x0A;

Data=Lc+1;
pwAppID = Data;
*pwAppID = BigEndianWORD(wAppID);
Data+=2;
memcpy(Data,"12345678",8);

Le= Data +8;
*Le=0x00;
Le++;
*Le=0x02;

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
```

```
/*关闭名称为"CAAPP"应用*/
*CLA=0x80;
*INS=0x28;
*P1=0x00;
*P2=0x00;
Lc=P2+1;
*Lc=0x00;

Lc++;
*Lc=0x00;
Lc++;
*Lc=0x02;

Data=Lc+1;
pwAppID= Data
*pwAppID =BigEndianWORD(wAppID);

RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
```

11. ECC 证书制作流程

```
/*以下编程范例演示在已存在的名称为"CAAPP"应用和名称为"12345678"的密钥容器条件下，如何生成 ECC 签名密钥对、如何导入 ECC 加密密钥对、如何导入数字证书等功能的实现*/
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
BYTE SW1,SW2;
BYTE CommAPDU[3000],RespAPDU[3000];
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;
WORD *pwAppID;
WORD wAppID;
WORD *pwContainerID;
```

```
WORD wContainerID;
WORD *pulBits;

CLA=CommAPDU;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
memset(CommAPDU,0,64);
memset(RespAPDU,0,64);

/*此处请参考其它章节的范例代码进行如下操作//
打开名称为"CAAPP"应用,并获取应用 ID
……
调用 VerifyPin 进行用户 PIN 码认证 */
……

/*打开已存在的容器名称为"12345678" 密钥容器*/
*CLA=0x80;
*INS=0x42;
*P1=0x00;
*P2=0x00;

Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0x00;
Lc++;
*Lc=0x0A;

Data=Lc+1;
pwAppID = Data;
*pwAppID = BigEndianWORD(wAppID);
Data+=2;
memcpy(Data,"12345678",8);

Le= Data +8;
*Le=0x00;
Le++;
*Le=0x02;

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
```

```
    /*错误处理*/
    ……
  }
  SW1=RespAPDU[RespLen-2];
  SW2=RespAPDU[RespLen-1];
  if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
  {
    /*错误处理*/
    ……
  }

  pwContainerID = RespAPDU;
  wContainerID = *pwContainerID;//此处需将 wContainerID 调整为机器字节序

  /*调用 GenECCKeyPair 生成 ECC 签名密钥对*/
  *CLA=0x80;
  *INS=0x70;
  *P1=0x00;
  *P2=0x00;

  Lc=P2+1;
  *Lc=0x00;
  Lc++;
  *Lc=0x00;
  Lc++;
  *Lc=0x06;

  Data=Lc+1;
  pwAppID = Data;
  *pwAppID =BigEndianWORD(wAppID);
  Data+=2;
  pwContainerID = Data;
  * pwContainerID = BigEndianWORD(wContainerID);
  Data+=2;
  pulBits = Data;
  * pulBits = BigEndianWORD(256);// 期望生成密钥对的位长度

  Le= Data +2;
  *Le=0x00;
  Le++;
  *Le=0x40; //期望返回的公钥数据字节长度

  RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
```

```
if (RetValue ! = SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /* 错误处理 */
  ……
}

/* 此处请将 ECC 公钥的 XCoordinate 和 YCoordinate 传递给 CA 系统,由 CA 系统生成 ECC 加密密钥对和对应的签名证书、加密证书,并由 CA 系统对加密密对按照格式要求进行组合。 */
……

/* 调用 ImportECCKeyPair 导入加密密钥对 */
* CLA=0x80;
* INS=0x72;
* P1=0x00;
* P2=0x00;

Lc=P2+1;
* Lc=0x00;
Lc++;
* Lc=0xXX; //数据长度高位字节
Lc++;
* Lc=0xXX; //数据长度低位字节

/* 导入的加密密钥对数据 */
Data=Lc+1;
memcpy(data, pECCKeyPair, ECCKeyPairLength);

RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
```

```
    {
      /* 错误处理 */
      ……
    }

    /* 导入签名数字证书 */
    * CLA=0x80;
    * INS=0x4C;
    * P1=0x00;
    * P2=0x00;

    Lc=P2+1;
    * Lc=0x00;
    Lc++;
    * Lc=0xXX; //应用 ID、容器 ID 和签名证书数据报文数据总长度的高位字节
    Lc++;
    * Lc=0xXX; //应用 ID、容器 ID 和签名证书数据报文数据总长度的低位字节

    Data=Lc+1;
    pwAppID = Data;
    * pwAppID = BigEndianWORD(wAppID);
    Data+=2;
    pwContainerID = Data;
    * pwContainerID = BigEndianWORD(wContainerID);
    Data+=2;
    *(BYTE*)Data = 1;//先导入签名证书
    Data+=1;
    *(ULONG*)Data =BigEndianDWORD(0xXX);//签名证书数据长度,需按照 BigEndian 方式
编码
    Data+=4;

    memcpy(Data,pSignCertData, signDataLength);// 签名数字证书数据

    RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
    if (RetValue != SAR_OK)
    {
      /* 错误处理 */
      ……
    }
    SW1=RespAPDU[RespLen-2];
    SW2=RespAPDU[RespLen-1];
    if ((SW1 != 0x90)||(SW2 != 0x00))
    {
```

```
  /* 错误处理 */
  ……
}


/* 导入加密数字证书 */
Lc=P2+1;
*Lc=0x00;
Lc++;
*Lc=0xXX; //应用 ID、容器 ID 和加密证书数据报文数据总长度的高位字节
Lc++;
*Lc=0xXX; //应用 ID、容器 ID 和加密证书数据报文数据总长度的低位字节

Data=Lc+1;
pwAppID = Data;
*pwAppID = BigEndianWORD(wAppID);
Data+=2;
pwContainerID = Data;
* pwContainerID = BigEndianWORD(wContainerID);
Data+=2;
*(BYTE*)Data = 0;//表示导入加密证书
Data+=1;
*(ULONG*)Data = BigEndianDWORD(0xXX);//加密证书数据长度,以 BigEndian 方式编码
Data+=4;
memcpy(Data,pEncCertData, encdataLength);// 加密数字证书数据

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /* 错误处理 */
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /* 错误处理 */
  ……
}

/* 关闭容器 */
*CLA=0x80;
*INS=0x44;
```

```
    *P1=0x00;
    *P2=0x00;

    Lc=P2+1;
    *Lc=0x00;
    Lc++;
    *Lc=0x00;
    Lc++;
    *Lc=0x04;

    Data=Lc+1;
    pwAppID = Data;
    *pwAppID = BigEndianWORD(wAppID);
    Data+=2;
    pwContainerID = Data;
    * pwContainerID = BigEndianWORD(wContainerID);

    RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
    if (RetValue != SAR_OK)
    {
      /*错误处理*/
      ……
    }
    SW1=RespAPDU[RespLen-2];
    SW2=RespAPDU[RespLen-1];
    if ((SW1 != 0x90)||(SW2 != 0x00))
    {
      /*错误处理*/
      ……
    }

    /*关闭名称为"CAAPP"应用*/
    *CLA=0x80;
    *INS=0x28;
    *P1=0x00;
    *P2=0x00;

    Lc=P2+1;
    *Lc=0x00;
    Lc++;
    *Lc=0x00;
    Lc++;
    *Lc=0x02;
```

```
Data=Lc+1;
pwAppID= Data
*pwAppID = BigEndianWORD(wAppID);

RetValue = SKF_Transmit( hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
```

12. 使用 SM2 密钥对进行数字签名

```
/*
*以下编程范例演示使用 SM2 密钥对进行数字签名
*/
#define ECC_PUBLIC_LEN XXXX //XXXX 为 SM2 密钥对长度
/* XX=1 或者 XX=2 当为 1 时输入为待签名数据的原文,当为 2 时输入的待签名数据为原始数据的杂凑值。*/
#define ECC_SIGNATURE_MODLE XX
BYTE *CLA,*INS,*P1,*P2,*Lc,*Data,*Le;
BYTE SW1,SW2;
BYTE CommAPDU[256],RespAPDU[256];
BYTE *Txt="GFA"; /*数据*/
ULONG CommLen;
ULONG DataLen;
ULONG RespLen;
ULONG RetValue;
BYTE Sign[256];
CLA=CommAPDU+0;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;
/*初始化命令缓冲区及返回数据的缓冲区*/
memset(CommAPDU,0, 256);
```

```
memset(RespAPDU,0, 256);
memset(Sign,0,256);
CLA[0]=0x80;
INS[0]=0x74;
P1[0]= ECC_SIGNATURE_MODLE;
P2[0]=0x00;
if(ECC_SIGNATURE_MODLE==0x01)
{
  DataLen=strlen(Txt);
  memcpy(Data, Txt, DataLen);
}
else(ECC_SIGNATURE_MODLE==0x02)
{
  /*对明文进行杂凑计算,将结果直接放进命令缓冲区中*/
  DataLen=HashCal(Txt , Data); /*HashCal 为杂凑计算,输入 Txt,输出到 Data,返回长度。*/
}
Lc[0]|=0xFF&(DataLen>>16);
Lc[1]|=0xFF&(DataLen>>8);
Lc[2]|=0xFF&( DataLen);
Le= CommAPDU +0x07+ DataLen;
Le[0]=0x00;
Le[1]=0x00;

CommLen=0x07+ DataLen+2; /*命令头+Lc 为 7 字节*/
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue ! = SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 ! = 0x90)||(SW2 ! = 0x00))
{
  /*错误处理*/
  ……
}
memcpy(Sign,RespAPDU,RespLen-2);
```

13. 使用 SM2 进行数字签名验证

```
#define ECC_PUBLIC_LEN XXXX          //SM2 密钥对位长度
BYTE *CLA, *INS, *P1, *P2, *Lc, *Data, *Le;
BYTE SW1,SW2;
```

```
BYTE CommAPDU[256],RespAPDU[256];
BYTE * Txt="GFA"
ULONG CommLen;
ULONG RespLen;
ULONG RetValue;
ULONG DataLen;
BYTE Sign[128];
ULONG ulBits = ECC_PUBLIC_LEN;
Byte x[ECC_PUBLIC_LEN/8];
Byte y[ECC_PUBLIC_LEN/8];
ULONG pbdata_len =0xXX;              //验签数据长度
Byte pbdata[0xXX];              //待验签数据,长度为验签数据长度
Byte r[ECC_PUBLIC_LEN/8];
Byte s[ECC_PUBLIC_LEN/8];

/*给待签名数据赋值*/
Memset(pbdata,0,128);
memcpy(pbdata, Txt, pbdata_len);
CLA=CommAPDU+0;
INS=CommAPDU+1;
P1=CommAPDU+2;
P2=CommAPDU+3;
Lc= CommAPDU+4;
Data= CommAPDU+7;
/*初始化*/
memset(CommAPDU,0,256);
memset(RespAPDU,0,256);
CLA[0]=0x80;
INS[0]=0x76;
P1[0]=0x00;
P2[0]=0x00;

/*给 Data 赋值*/
DataLen=0;              //初始化数据长度为 0
Data[0]|=0xFF&( ulBits>>24);
Data[1]|=0xFF&( ulBits>>16);
Data[2]|=0xFF&( ulBits>>8);
Data[3]|=0xFF&( ulBits);
DataLen+=4;
memcpy(Data+ DataLen, x, ECC_PUBLIC_LEN/8);
DataLen+= ECC_PUBLIC_LEN/8;
memcpy(Data+ DataLen, y, ECC_PUBLIC_LEN/8);
DataLen+= ECC_PUBLIC_LEN/8;
```

```
Data[Data+ DataLen+0]|= 0xFF&(pbdata_len>>24);
Data[Data+ DataLen+1]|= 0xFF&(pbdata_len>>16);
Data[Data+ DataLen+2]|= 0xFF&(pbdata_len>>8);
Data[Data+ DataLen+3]|= 0xFF&(pbdata_len>>0);
DataLen+=4;
memcpy(Data+ DataLen, pbdata, pbdata_len);
DataLen+= pbdata_len;
memcpy(Data+ DataLen, r, ECC_PUBLIC_LEN/8);
DataLen+= ECC_PUBLIC_LEN/8;
memcpy(Data+ DataLen, s, ECC_PUBLIC_LEN/8);
DataLen+= ECC_PUBLIC_LEN/8;
Lc[0]|=0xFF&(DataLen>>16);
Lc[1]|=0xFF&(DataLen>>8);
Lc[2]|=0xFF&(DataLen>>0);

CommLen=0x07+ DataLen; /*命令头+Lc为7字节*/
RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
```

14. 使用SM2密钥对交换会话密钥

```
//发起方:
uchar CommAPDU[256],RespAPDU[256];
struct Apdu_ecc{
  struct Head{
    uchar CLA;
    uchar INS;
    uchar P1;
    uchar P2;
    }Head;
    uchar LC1= CommAPDU[4];
    uchar LC2= CommAPDU[5];
    uchar LC3= CommAPDU[6];
```

```
    }Apdu_ecc;
    ULONG secret_handler;/*会话密钥句柄*/
    uchar sendAPDU[256];

    ushort lc;
    unsigned long privatetmpkey;
    memset(CommAPDU,0, 256);
    memset(RespAPDU,0, 256);
    privatetmpkey= generationEccRand();
    /*
    由 privatetmpkey 生成响应方临时 ECC 公钥 ECCPUBLICKEYBLOB
    */

    lc=4+4+ulResponserBits/8+ulResponserBits/8+4+ResponserTempXCoordinate/8+Respon-
serTempXCoordinate/8+4+ulResponserIDLen;
    //设置指令信息
    Apdu_ecc.CLA=0x80;
    Apdu_ecc.INS=0x84;
    Apdu_ecc.P1=0x00;
    Apdu_ecc.p2=0x00;
    Apdu_ecc.LC1=0;
    Apdu_ecc.LC2= 0xFF &(lc>>8);
    Apdu_ecc.LC3= 0xFF &(lc);
    memcpy(CommAPDU,&Apdu_ecc,7);/*把 APDU 头放到 CommAPDU 里*/
    /*
    将对应的 key 数据,及长度等填入 CommAPDU 的数据段中。
    */
    RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
    if (RetValue != SAR_OK)
    {
      /*错误处理*/
      ……
    }
    SW1=RespAPDU[RespLen-2];
    SW2=RespAPDU[RespLen-1];
    if ((SW1 != 0x90)||(SW2 != 0x00))
    {
      /*错误处理*/
      ……
    }
    memcpy(&secret_handler,RespAPDU,RespLen-2);

    }
```

```
//接收方：
struct Apdu_ecc{
  struct Head{
    uchar CLA;
    uchar INS;
    uchar P1;
    uchar P2;
    }Head;
    uchar LC1;
    uchar LC2;
    uchar LC3;
  }Apdu_ecc;
ULONG secret_handler;/*会话密钥句柄*/
uchar recvAPDU[256];

ushort lc;
unsigned long privatetmpkey;
uchar CommAPDU[256],RespAPDU[256];
memset(CommAPDU,0, 256);
memset(RespAPDU,0, 256);

recvapdu(recvAPDU); /*从发起方获取 APDU*/
if(recvAPDU[0]==0x80 && recvAPDU[1]==0x84 * * recvAPDU[2]==0x00 && recvAPDU[3]== 0x00)/*判断是发起方发送的用于协商 ECC 密钥的 PADU*/
{
/*
保存发起方 APDU 里的对应信息
*/

  privatetmpkey= generationEccRand();
/*
由 privatetmpkey 生成响应方临时 ECC 公钥 ECCPUBLICKEYBLOB
*/

lc=4+4+ulResponserBits/8+ulResponserBits/8+4+ResponserTempXCoordinate/8+ResponserTempXCoordinate/8+4+ulResponserIDLe;
//设置指令信息
Apdu_ecc.CLA=0x80;
Apdu_ecc.INS=0x86;
Apdu_ecc.P1=0x00;
Apdu_ecc.P2=0x00;
Apdu_ecc.LC1=0;
Apdu_ecc.LC2= 0xFF &(lc>>8);
```

```
Apdu_ecc.LC3= 0xFF &(lc);

memcpy(CommAPDU,&Apdu_ecc,7);//把 APDU 头放到 CommAPDU 里

/*
将对应的 key 数据,及长度等填入 CommAPDU 的数据段中。
*/

RetValue = SKF_Transmit(hDev,CommAPDU,CommLen,RespAPDU,&RespLen);
if (RetValue != SAR_OK)
{
  /*错误处理*/
  ……
}
SW1=RespAPDU[RespLen-2];
SW2=RespAPDU[RespLen-1];
if ((SW1 != 0x90)||(SW2 != 0x00))
{
  /*错误处理*/
  ……
}
memcpy(&secret_handler,RespAPDU,RespLen-2);

}
```