ICS 35.040
L 80
备案号：

中华人民共和国密码行业标准

GM/T 0056-2018

多应用载体密码应用接口规范

Specification of cryptography application interface with multi-applications equipment

（报批稿）

2018-05-02 发布　　2018-05-02 实施

国家密码管理局　发布

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。

本标准由密码行业标准化技术委员会提出并归口。

本标准起草单位：北京中电华大电子设计有限责任公司、上海华虹集成电路有限责任公司、北京同方微电子有限公司、恒宝股份有限公司、北京握奇数据系统有限公司、东信和平科技股份有限公司、北京华大智宝电子系统有限公司、上海复旦微电子集团股份有限公司、国民技术股份有限公司、北京南瑞智芯微电子科技有限公司、成都信息工程大学、武汉天喻信息产业股份有限公司、华大半导体有限公司。

本标准主要起草人：兰天、吴秉男、苑中魁、袁巧、陈操、刘平、王庆林、王怀英、耿佳、白长虹、汪雪琳、张楠、王永吉、李志远、陈悦、李静进、何迪、赵永刚、王宝鸫、陈安新、吴震、饶金涛、黄惠瑜、许晶、刘欣。

引　言

本文中多应用载体是指具备独立、开放的片上操作系统、提供多应用运行环境、支持载体上多个应用的下载、安装、重用、共存和安全隔离的载体，通常由硬件、驱动、COS和应用构成。

多应用载体中的用户应用在使用SM2/3/4系列算法时，需要载体的多应用环境提供SM2/3/4系列算法的密码应用调用接口。由于目前多应用载体相关标准未定义SM2/3/4系列算法的应用接口，造成用户应用无法使用的问题。为此，编制本标准以规范SM2/3/4系列算法在多应用载体中的密码算法能力标识、接口规格，保障用户应用使用密码功能的统一性和完整性。

多应用载体可以使用不同的技术方案实现，如Java技术方案、C技术方案等。本版本主要描述了Java技术方案中的密码应用接口，其他技术方案的密码应用接口根据应用发展情况在后续版本中给出。

多应用载体密码应用接口规范

1 范围

本标准规定了多应用载体中SM2/3/4系列算法的密码应用接口，包括：

--定义SM2/SM3/SM4算法在多应用载体中的标识。

--定义SM2/SM3/SM4的算法的密码应用接口规格。

本标准适用于各种多应用载体的研制，也可用于指导多应用载体的密码应用检测。

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。

GB/T 32905-2016 信息安全技术 SM3密码杂凑算法

GB/T 32907-2016 信息安全技术 SM4分组密码算法

GB/T 32918-2016 信息安全技术 SM2椭圆曲线公钥密码算法

RFC 2898 Specification of PKCS #5

3 术语和定义

下列术语和定义适用于本文件。

3.1

命令 command

终端向载体发出的一条信息，该信息启动一个操作或请求一个应答。

3.2

响应 response

载体处理完成收到的命令报文后，回送给终端的报文。

3.3

报文 message

由终端向载体或载体向终端发出的，不含传输控制字符的字节串。

3.4

多应用载体 multi-applications equipment

本文中指具备独立、开放的片上操作系统、提供多应用运行环境、支持载体上多个应用的下载、安装、重用、共存和安全隔离的载体，通常由硬件、驱动、COS和应用构成。

3.5

SM2算法 SM2 algorithm
由GB/T 32918定义的一种算法。

3.6

SM3算法 SM3 algorithm
由GB/T 32905定义的一种算法。

3.7

SM4算法 SM4 algorithm
由GB/T 32907定义的一种算法。

3.8

SM9算法 SM9 algorithm
一种采用双线性对的椭圆曲线公钥密码算法。

4 符号和缩略语

下列符号和缩略语适用于本文件。

AID 应用标识符（Application Identifier）
AKEY 辅助密钥（Auxiliary Key）
API 应用编程接口（Application Programming Interface）
CBC 链式加密（Cipher-block chaining）
COS 片上操作系统（Chip Operating System ）
ECB 电子密码本（Electronic Codebook）
ISO 国际标准化组织（International Organization for Standardization）
MAC 消息鉴别码（Message Authentication Code ）
MKEY 报文密钥（Message Key）
OFB 输出反馈（Output Feedback）

5 多应用载体系统框架

多应用载体由硬件层、驱动层、OS层、应用层构成，详见图1。

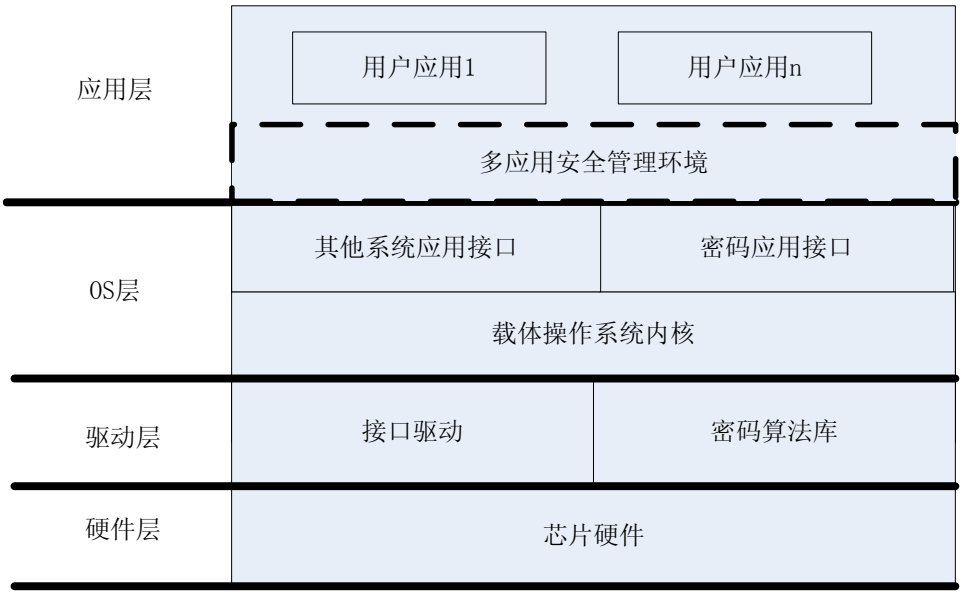

图1　多应用载体安系统框架

其中硬件层是芯片，包括CPU、密码算法协处理器、存储器等。

驱动层包括各芯片硬件的接口驱动、以及和密码算法协处理器配套的算法库。

OS层包括支持多应用调度的载体操作系统内核,向应用层提供的系统应用接口,以及密码应用接口。

应用层包括多应用安全管理环境以及各种用户应用。多应用安全管理环境是一个特定的用户级别程序，用于实现多应用的安全管理和安全隔离；用户应用实现各种业务应用功能。应用层通过OS层的密码应用接口实现密码功能的调用。

6　多应用载体密码应用接口调用流程

6.1　密码应用接口调用流程

多应用载体中，调用密码应用接口的流程如图2所示：

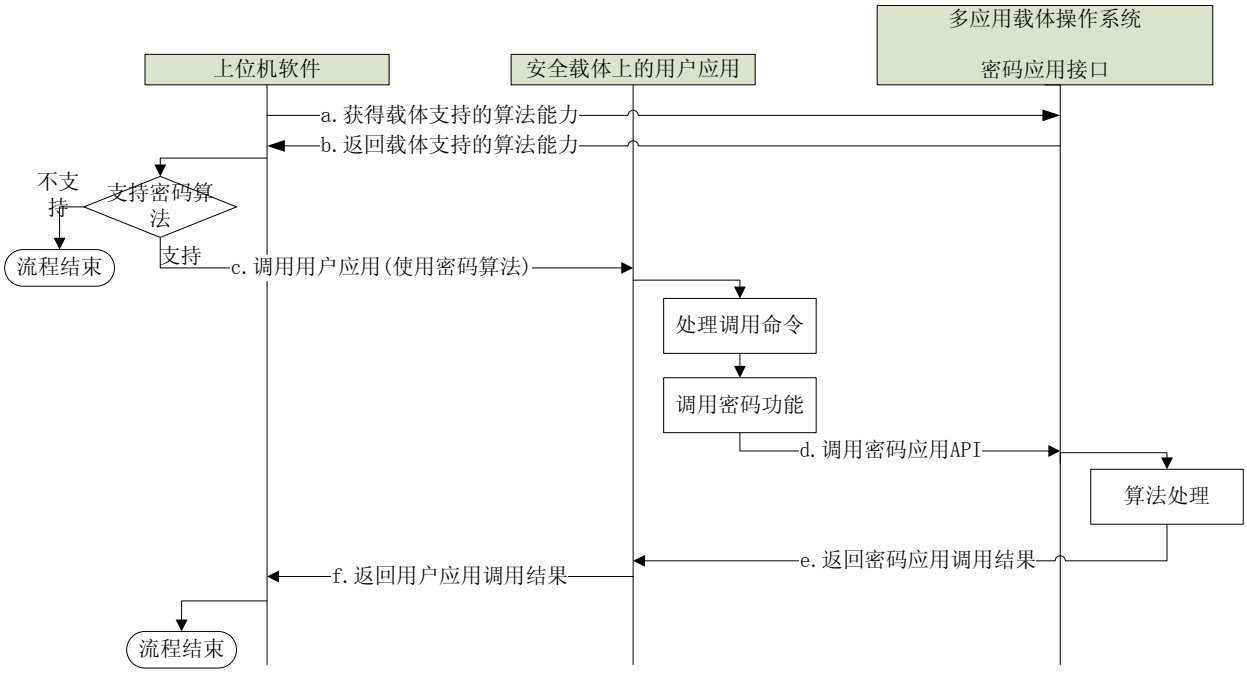

图2　多应用载体密码应用接口调用流程

上位机软件是载体外的客户端软件，用户应用是在安全载体内运行的、实现密码安全功能嵌入式软件。

用户应用调用密码算法的流程说明如下：

a）获得载体算法能力：上位机软件使用多应用载体的基本命令，获取载体上的算法能力，该命令和载体上的用户应用无关。该步骤为可选步骤，如果上位机软件已知该载体的算法能力，可以跳过该步骤。

b）返回算法能力：多应用载体操作系统返回载体支持的算法能力。如果载体算法能力和上位机不匹配，流程结束。如果匹配，转c。该步骤为可选步骤，如果客户端软件已知该载体的算法能力，可以跳过该步骤。

c）调用密码应用接口：如果载体支持密码算法，上位机软件调用载体上的用户应用。载体上用户应用收到调用命令后，根据命令进行处理。

d）调用密码应用接口：如果该命令的处理需要使用密码功能，通过载体操作系统的密码应用接口进行调用。

e）返回密码应用接口调用结果：密码应用API被调用后，进行算法处理并向载体上的用户应用返回处理结果。

f）返回应用调用结果：载体上的用户应用向上位机软件返回调用结果。流程结束。

6.2　密码算法能力标识

多应用载体应对所支持的密码算法能力进行标识，以便于载体外实体获得载体算法支持情况。

由于多应用载体可以使用不同的技术方案实现，如Java技术方案、C技术方案等，相应技术方案的算法能力标识参见第7章。

6.3　密码应用接口规格

多应用载体中应定义密码应调用的接口规格，以便于载体上的用户应用调用。

由于安全载体在实现中使用不同的技术方案，如Java技术方案、C技术方案等，相应技术方案的调用接口规格参见第7章。

7 Java 技术方案密码应用接口

7.1 简介

本部分描述了使用Java技术方案的多应用载体中，SM2/3/4系列算法的能力标识和应用接口规格定义，使用SM2/3/4系列算法的应用可以调用这些应用接口以达到对密码功能的使用。

密码算法中的密钥对象需通过GMKeyBuilder.buildKey方法创建密钥实例，然后通过setXXX等方法设置密钥对象所使用的密钥值，签名算法、加解密算法将使用这些密钥对象。对称密钥对象包括：SM4Key，非对称密钥对象包括：SM2PrivateKey、SM2PublicKey。

签名与验证算法是通过GMSignature类实现的，在使用签名与验证算法前需要通过GMSignature.getInstance方法获得相应算法的实例对象，再配合相应的密钥对象以实现生成签名数据与验证签名数据的目的。

数据加解密算法是通过GMCipher类实现的，在使用数据加解密算法前需要通过GMCipher.getInstance方法获得相应算法的实例对象,再配合相应的密钥对象以实现对数据加密或解密的目的。

数据杂凑算法是通过GMMessageDigest类实现的，在使用数据杂凑算法前需要通过GMMessageDigest.getInstance方法获得SM3算法的实例对象，然后才可对数据进行杂凑计算。

密码算法API中定义了GMKeyPair类，用于在载体内生成密码算法中的非对称密钥对（SM2）。

7.2 密码算法能力标识

多应用载体密码算法能力标识定义见表1。

表1 密码算法能力标识

标识	对应密码算法能力
0x00 - 0x0F	保留
0x10 - 0x17	保留
0x18	SM4_ECB
0x19	SM4_CBC
0x1A	SM4_OFB
0x1B	保留
0x1C - 0x1F	保留
0x20	SM3_256
0x21 - 0x27	保留
0x28	SM2_256
0x29 - 0x2F	保留
0x30	SM9_256
0x31 - 0xFF	保留

7.3 密码应用包定义

密码应用包定义见表2。

表2 包信息

项	定义
包名	com.guomi
AID	0xA0:0x0:0x0:0x0:0x0:0x0:0x53:0x43:0x43:0x41:0x01
主版本号	1
次版本号	0

7.4 密码应用接口定义

7.4.1 类定义

密码应用类定义如下。

class java.lang.Object

——class com.guomi.**GMCipher**

——class com.guomi.**GMKeyBuilder**

——class com.guomi.**GMKeyPair**

——class com.guomi.**GMMessageDigest**

——class com.guomi.**GMSignature**

——class com.guomi.**GMCipherExtend**

——class com.guomi.**GMSM2KeyExchange**

7.4.2 接口定义

密码应用接口定义如下。

interface javacard.security.Key

——interface javacard.security.SecretKey

——interface com.guomi.**SM4Key**

——interface javacard.security.PrivateKey

——interface com.guomi.**SM2PrivateKey**

——interface javacard.security.PublicKey

——interface com.guomi.**SM2PublicKey**

interface com.guomi.SM2Key

——interface com.guomi.**SM2PrivateKey**

——interface com.guomi.**SM2PublicKey**

7.5 密码应用类信息

7.5.1 SM4Key

7.5.1.1 概述

SM4Key包含一个16字节的密钥AKEY，用于SM4算法的加解密运算。

一旦密钥数据被设置，该密钥对象就将处于初始化完成状态（isInitialized方法将返回true），该密钥可以被使用了。

SM4Key接口定义见表3。

表3 SM4Key

返回值	定义	说明
byte	getKey(byte keyData[], short kOff)	获得密钥引用
void	setKey(byte keyData[], short kOff)	设置密钥数据

7.5.1.2 setKey

7.5.1.2.1 声明

```
public abstract void setKey(
    byte keyData[],
    short kOff
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.1.2.2 描述

本方法用于设置SM4密钥数据。输入的密钥数据长度为16字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.1.2.3 参数

keyData - 此byte数组存放待设置的密钥数据。
kOff - 密钥数据在byte数组中的起始位置。

7.5.1.2.4 抛出异常

CryptoException.ILLEGAL_VALUE - 密码数据需要解密而且解密出错时抛出该异常。
NullPointerException - keyData参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - kOff参数为负数或者超过keyData数组时，或者kOff加上密钥数据长度超出keyData数据长度时，抛出该异常。

7.5.1.3 getKey

7.5.1.3.1 声明

```
public abstract byte getKey(
    byte keyData[],
    short kOff
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.1.3.2 描述

本方法获得的SM4密钥引用。密钥引用指向的数据长度为16字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.1.3.3 参数

keyData - 此byte数组存放返回的密钥数据。
kOff - 密钥数据在byte数组中的起始位置。

7.5.1.3.4 返回值

以字节计的密钥数据的长度。

7.5.1.3.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥对象没有正确初始化时。
NullPointerException - keyData参数为null时。
ArrayIndexOutOfBoundsException - kOff参数为负数或者超过keyData数组时，或者kOff加上密钥数据长度超出keyData数据长度时，抛出该异常。

7.5.2 SM2Key

7.5.2.1 概述

SM2Key是非对称密钥的一种，分为公钥和私钥，用于SM2算法的运算。
SM2Key接口定义见表4。

表4 SM2Key

返回值	定义	说明
short	getA(byte[] buffer, short offset)	获得曲线参数A
short	getB(byte[] buffer, short offset)	获得曲线参数B
short	getG(byte[] buffer, short offset)	获得曲线参数G

7.5.2.2 getA

7.5.2.2.1 声明

```
public short getA(
  byte buffer[],
  short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException ;
```

7.5.2.2.2 描述

本方法用于获取曲线参数A的数据。输出的数据长度为32字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.2.2.3 参数

buffer - 此byte数组存放输出的数据。
offset - 数据在byte数组中的起始位置。

7.5.2.2.4 返回值

以字节计的曲线参数A的数据长度。

7.5.2.2.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥处于未初始化状态时抛出该异常。
NullPointerException - keyData参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上曲线参数A长度超出buffer数据长度时，抛出该异常。

7.5.2.3 getB

7.5.2.3.1 声明

```
public short getB(
  byte buffer[],
  short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.2.3.2 描述

本方法用于获取曲线参数B的数据。输出的数据长度为32字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.2.3.3 参数

buffer - 此byte数组存放输出的数据。
offset - 数据在byte数组中的起始位置。

7.5.2.3.4 返回值

以字节计的曲线参数B的数据长度。

7.5.2.3.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥处于未初始化状态时抛出该异常。
NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上曲线参数B长度超出keyData数据长度时，抛出该异常。

7.5.2.4 getG

7.5.2.4.1 声明

```
public short getG(
  byte buffer[],
  short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.2.4.2 描述

本方法用于获取曲线参数G的数据。输出的数据长度为64字节,前32字节为G值x坐标，后32字节为y坐标。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.2.4.3 参数

buffer - 此byte数组存放输出的数据。
offset - 数据在byte数组中的起始位置。

7.5.2.4.4 返回值

以字节计的曲线参数G的数据长度。非压缩格式，返回的G值格式为x坐标和y坐标内容，前面不包含标识信息。

7.5.2.4.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥处于未初始化状态时抛出该异常。
NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上曲线参数G长度超出buffer数据长度时，抛出该异常。

7.5.3 SM2PrivateKey

7.5.3.1 概述

SM2PrivateKey接口用于生成数据的签名。
由于SM2算法曲线方程和参数已经固定，一旦私钥数据被设置，该私钥对象就将处于初始化完成状态（isInitialized方法将返回true），该密钥可以被使用了。
SM2PrivateKey接口定义见表5。

表5 SM2PrivateKey

返回值	定义	说明
short	getS(byte[] buffer, short offset)	获得私钥数据
void	setS(byte[] buffer, short offset, short length)	设置私钥数据

7.5.3.2 getS

7.5.3.2.1 声明

```
public short getS(
  byte[] buffer,
  short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.3.2.2 描述

本方法获得的SM2私钥数据。输出的私钥引用数据长度为32字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.3.2.3 参数

buffer - 此byte数组存放返回的私钥数据。
offset - 数据在byte数组中的起始位置。

7.5.3.2.4 返回值

以字节计的私钥数据的长度，本规范中固定为32。

7.5.3.2.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥处于未初始化状态时抛出该异常。
NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上密钥数据长度超出buffer数据长度时，抛出该异常。

7.5.3.3 setS

7.5.3.3.1 声明

```
public void setS(
  byte[] buffer,
  short offset
  short length
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.3.3.2 描述

本方法用于设置SM2私钥数据。输入的私钥引用数据长度为32字节。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.3.3.3 参数

buffer - 此byte数组存待设置的私钥数据。
offset - 数据在byte数组中的起始位置。
length - 以字节计的私钥数据的长度，本规范中固定为32。

7.5.3.3.4 抛出异常

CryptoException.ILLEGAL_VALUE - 输入的密钥长度不符合要求时或者密钥需要解密并且出错时抛出该异常。
NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上密钥数据长度超出buffer数据长度时，抛出该异常。

7.5.4 SM2PublicKey

7.5.4.1 概述

SM2PublicKey接口用于验证数据的签名。
因为曲线方程和参数已经固定，因此一旦公钥数据被设置，该公钥对象就将处于初始化完成状态（isInitialized方法将返回true），该公钥可以被使用。
SM2PublicKey接口定义见表6。

表6 SM2PublicKey

返回值	定义	说明
short	getW(byte[] buffer, short offset)	获得公钥数据
void	setW(byte[] buffer, short offset, short length)	设置公钥数据

7.5.4.2 getW

7.5.4.2.1 声明

```
public short getW(
  byte[] buffer,
  short offset
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.4.2.2 描述

本方法获得的SM2公钥数据。输出的公钥引用数据长度为64字节，前32字节为公钥x坐标，后32字节为y坐标。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。

7.5.4.2.3 参数

Buffer - 此byte数组存放返回的公钥数据。
offset - 数据在byte数组中的起始位置。

7.5.4.2.4 返回值

公钥数据的长度为 64字节。公钥数据格式为x坐标和y坐标内容，前面不包含标识信息。

7.5.4.2.5 抛出异常

CryptoException.UNINITIALIZED_KEY - 密钥处于未初始化状态时抛出该异常。
NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上密钥数据长度超出buffer数据长度时，抛出该异常。

7.5.4.3 setW

7.5.4.3.1 声明

```
public void setW(
  byte[] buffer,
  short offset
  short length
)throws CryptoException, NullPointerException, ArrayIndexOutOfBoundsException;
```

7.5.4.3.2 描述

本方法用于设置SM2公钥数据。输入的公钥引用数据长度为64字节，前32字节为公钥x坐标，后32字节为y坐标。数据格式为大端并且右对齐，即最低有效位为最后那个字节的最低有效位。。

7.5.4.3.3 参数

buffer - 此byte数组存待设置的公钥数据。

offset　- 数据在byte数组中的起始位置。

Length　- 公钥数据的长度，以字节计，必须为64。

此处的输入数据格式为x坐标和y坐标内容，前面不包含标识信息。

7.5.4.3.4　抛出异常

CryptoException.ILLEGAL_VALUE　- 输入的密钥长度不符合要求时抛出该异常。

NullPointerException　- buffer参数为null时抛出该异常。

ArrayIndexOutOfBoundsException　- offset参数为负数或者超过buffer数组时，或者offset加上密钥数据长度超出buffer数据长度时，抛出该异常。

7.5.5　GMKeyPair

7.5.5.1　概述

GMKeyPair类是一个密钥对（包括一个公开密钥和一个私有密钥）的容器。创建之后，容器中的公钥和私钥都处于未初始化状态。

GMKeyPair类定义见表7。

表7　GMKeyPair

类型	定义	说明
public static final byte	ALG_SM2_FP = 127	生成密钥对的类型为素域的SM2密钥
	GMKeyPair(byte algorithm ,short keyLength)	为指定的密钥长度创建一个GMKeyPair实例；封装的密钥处于未初始化状态 // keyLength确定长度
	GMKeyPair(PublicKey public_key,PrivateKey private_key)	创建一个包含指定公钥和私钥的新的GMKeyPair对象
PublicKey	getPublic()	返回对这个GMKeyPair对象中公钥组件的引用
PrivateKey	getPrivate()	返回对这个GMKeyPair对象中私钥组件的引用
Void	genSM2KeyPair ()	用新密钥值（重新）初始化GMKeyPair实例中的密钥对象

7.5.5.2　GMKeyPair

7.5.5.2.1　声明

```
public GMKeyPair(
  byte algorithm,
  short keyLength) throws CryptoException;
```

7.5.5.2.2　描述

为指定的密钥长度构造一个GMKeyPair实例；被封装的密钥是未初始化的。使用genSM2KeyPair ()方法对GMKeyPair对象进行初始化。

被封装的密钥对象具有指定的keyLength并且实现了相关的算法的(比如,在一个ALG_SM2密钥对中，SM2PublicKey是公钥接口，SM2PrivateKey是私钥接口）Key接口。

7.5.5.2.3 参数

Algorithm - 生成密钥对的类型。
keyLength - 用位标识的密钥长度，有效密钥位数依赖于密钥类型。

7.5.5.2.4 抛出异常

CryptoException.NO_SUCH_ALGORITHM - 如果载体不支持输入的密钥对类型抛出该异常。

7.5.5.3 GMKeyPair

7.5.5.3.1 声明

```
public GMKeyPair(
  PublicKey public_key,
  PrivateKey private_key
) throws CryptoException;
```

7.5.5.3.2 描述

构造一个包含了指定公钥和私钥的新的GMKeyPair对象。

该构造器仅将公钥和私钥的引用存入生成的GMKeyPair对象中。如果密钥参数对象没有被初始化，不抛出异常。

7.5.5.3.3 参数

private_key - 私钥。
public_key - 公钥。

7.5.5.3.4 抛出异常

CryptoException.NO_SUCH_ALGORITHM - 如果密钥的类型和长度不支持时抛出该异常。

CryptoException.ILLEGAL_VALUE - 如果输入的密钥为空时或者公私钥密钥的类型和长度不匹配时抛出该异常。

7.5.5.4 genSM2KeyPair

7.5.5.4.1 声明

```
public final void genSM2KeyPair ()throws CryptoException;
```

7.5.5.4.2 描述

生成新的密钥值并重新初始化各个GMKeyPair实例中的密钥对象。此实例中已初始化的公钥和私钥与GMSignature和GMCipher对象配合使用。

7.5.5.4.3 抛出异常

CryptoException.ILLEGAL_VALUE - 当预设置的曲线参数A,B,G等不合法时，抛出该异常。（目前本规范A,B,G固定，不会出现该异常，该异常用于后续扩展）。

7.5.5.5 getPublic

7.5.5.5.1 声明

public PublicKey getPublic ();

7.5.5.5.2 描述

返回该GMKeyPair对象的公钥组件的引用。

7.5.5.5.3 返回值

公钥引用。

7.5.5.6 getPrivate

7.5.5.6.1 声明

public PrivateKey getPrivate ();

7.5.5.6.2 描述

返回此GMKeyPair对象的私钥组件引用。

7.5.5.6.3 返回值

私钥的引用。

7.5.6 GMSignature

7.5.6.1 概述

GMSignature类是密码算法签名算法的基类。密码算法签名算法的实现必须扩展这个类，并实现所有的抽象方法。

术语“填充”用在下面的公钥签名算法中是指在被参考的方案中将消息杂凑变换到加密块大小指定的所有操作。

GMSignature类定义见表8。

表8 GMSignature

类型	定义	说明
public static final byte	ALG_SM4_MAC4_NOPAD = 0x89	使用CBC模式的SM4产生一个4字节的MAC（被加密块的高位4字节），不对输入数据进行填充
public static final byte	ALG_SM4_MAC8_NOPAD = 0x8A	使用CBC模式的SM4产生一个8字节的MAC（被加密块的高位8字节），不对输入数据进行填充
public static final byte	ALG_SM4_MAC16_NOPAD = 0xAA	使用CBC模式的SM4产生一个16字节的MAC，不对输入数据进行填充
public static	ALG_SM4_MAC4_ISO9797_M1 = 0x8B	使用CBC模式的SM4产生一个4字节的MAC（被加

类型	定义	说明
final byte		密块的高位4字节），输入数据使用ISO9797M1算法填充
public static final byte	ALG_SM4_MAC8_ISO9797_M1 = 0x8C	使用CBC模式的SM4产生一个8字节的MAC（被加密块的高位8字节），输入数据使用ISO9797M1算法填充
public static final byte	ALG_SM4_MAC16_ISO9797_M1 = 0xAB	使用CBC模式的SM4产生一个16字节的MAC，输入数据使用ISO9797M1算法填充
public static final byte	ALG_SM4_MAC4_ISO9797_M2 = 0x8D	使用CBC模式的SM4产生一个4字节的MAC（被加密块的高位4字节），输入数据使用ISO9797M2算法填充
public static final byte	ALG_SM4_MAC8_ISO9797_M2 = 0x8E	使用CBC模式的SM4产生一个8字节的MAC（被加密块的高位8字节），输入数据使用ISO9797M2算法填充
public static final byte	ALG_SM4_MAC16_ISO9797_M2 = 0xAC	使用CBC模式的SM4产生一个16字节的MAC，输入数据使用ISO9797M2算法填充
public static final byte	ALG_SM4_MAC4_PKCS5 = 0x8F	使用CBC模式的SM4产生一个4字节的MAC（被加密块的高位4字节），输入数据使用PKCS5算法填充
public static final byte	ALG_SM4_MAC8_PKCS5 = 0x90	使用CBC模式的SM4产生一个8字节的MAC（被加密块的高位8字节），输入数据使用PKCS5算法填充
public static final byte	ALG_SM4_MAC16_PKCS5 = 0xAD	使用CBC模式的SM4产生一个16字节的MAC，输入数据使用PKCS5算法填充
public static final byte	ALG_SM2_SM3_256 = 0xA1	产生一个32字节的SM3杂凑,并使用SM2加密它，不对输入数据进行填充
public static final byte	ALG_SM2_SM3_256_INPUT_E = 0xB0	对输入的32字节的SM3杂凑,使用SM2计算签名
protected	GMSignature ()	受保护的构造方法
public static final Signature	getInstance (byte algorithm, boolean externalAccess)	创建选定GMSignature算法的签名对象的实例

注：1.getInstance()方法返回SM系列算法签名对象实例；该对象实例实现了标准Signature类的方法。

使用ALG_SM2_SM3_256算法的Signature对象，调用sign()方法生成签名时，inbuff的输入内容格式为（ZA|Message），sign()过程中对（ZA|Message）进行摘要算法，并生成签名。

使用ALG_SM2_SM3_256_INPUT_E算法的Signature对象，调用sign()方法生成签名时，inbuff的输入内容格式为32字节SM3结果，使用SM3算法对数据（ZA|Message）进行杂凑算法运算的结果，sign()过程中对SM3结果生成签名。

ZA的定义参考GB/T 32918的第2部分。

SM2的签名数据格式为64字节的r和s，前面不包含标识信息。验签数据格式也不包含标识信息。

7.5.6.2 GMSignature 构造方法

7.5.6.2.1 声明

```
protected GMSignature();
```

7.5.6.2.2 描述

GMSignature类不能够直接进行实例化，需要通过本构造器为扩展构造。

7.5.6.3 getInstance

7.5.6.3.1 声明

```
public static final Signature getInstance(
  byte algorithm,
  boolean externalAccess
) throws CryptoException;
```

7.5.6.3.2 描述

创建选定Signature算法的签名对象的实例。

7.5.6.3.3 参数

algorithm　　- 期望的GMSignature算法。上面在“GMSignature.ALG_”常量中所列出的有效编码，如ALG_DES_MAC4_NOPAD。

externalAccess　- true表示实例将在多个applet实例中共享，当GMSignature实例的所有者不是当前选定的applet时，GMSignature实例也可以被访问（通过一个Shareable接口）。如果值为true，那么实现必须不能为内部数据分配CLEAR_ON_DESELECT临时空间。

7.5.6.3.4 返回值

返回被请求算法的Signature对象实例。

7.5.6.3.5 抛出异常

CryptoException.NO_SUCH_ALGORITHM　- 如果被请求的算法或者被共享的访问模式不被支持抛出该异常。

7.5.7 GMMessageDigest

7.5.7.1 概述

GMMessageDigest类扩展自Object类，用于计算指定算法的消息杂凑。

GMMessageDigest类定义见表9。

表9 GMMessageDigest

类型	定义	说明
public static final byte	ALG_SM3_256 = 0x81	SM3算法，结果为256位
public static final byte	ALG_SM3_384 = 0x84	SM3算法，结果为384位

类型	定义	说明
public static final byte	ALG_SM3_512 = 0x85	SM3算法，结果为512位
public static final byte	LENGTH_SM3_256 = 32	256位SM3算法结果长度，32字节
public static final byte	LENGTH_SM3_384 = 48	384位SM3算法结果长度，48字节
public static final byte	LENGTH_SM3_512 = 64	512位SM3算法结果长度，64字节
Protected	GMMessageDigest ()	受保护的构造方法
public static final MessageDigest	getInstance (byte algorithm, boolean externalAccess)	根据指定的算法，创建对应的MessageDigest类的实例

注：1.getInstance()方法获取SM系列算法的摘要对象实例，该对象实例实现了标准MessageDigest类的接口。

7.5.7.2 GMMessageDigest 构造方法

7.5.7.2.1 声明

```
protected GMMessageDigest();
```

7.5.7.2.2 描述

GMMessageDigest类的构造器。

7.5.7.2.3 返回值

返回GMMessageDigest类的实例。

7.5.7.3 getInstance

7.5.7.3.1 声明

```
public static final MessageDigest getInstance(
  byte algorithm,
  boolean externalAccess
) throws CryptoException;
```

7.5.7.3.2 描述

创建对应的密码算法消息杂凑算法。

7.5.7.3.3 参数

algorithm　　- 消息杂凑算法值。

externalAccess　- 指定消息杂凑实例是否可被多个应用实例访问，true表明消息杂凑实例可被多个应用实例访问，false表明不可以被多个应用实例访问。

7.5.7.3.4 返回值

返回指定算法的消息杂凑实例。

7.5.7.3.5 抛出异常

CryptoException.NO_SUCH_ALGORITHM - 当指定的算法不支持时，抛出该异常。

7.5.8 GMKeyBuilder

7.5.8.1 概述

GMKeyBuilder类是密码算法的密钥工厂类，通过指定参数能够创建各种密码算法的密钥对象。GMKeyBuilder类定义见表10。

表10 GMKeyBuilder

类型	定义	说明
public static final byte	TYPE_SM4_TRANSIENT_RESET = 0x84	此类型密钥对象实现了SM4Key接口，且其密钥数据存储在CLEAR_ON_RESET空间
public static final byte	TYPE_SM4_TRANSIENT_DESELECT = 0x85	此类型密钥对象实现了SM4Key接口，且其密钥数据存储在CLEAR_ON_DESELECT空间
public static final byte	TYPE_SM4 = 0x86	此类型密钥对象实现了SM4Key接口，且其密钥数据存储在非易失性存储器中
public static final byte	TYPE_SM2_PUBLIC = 0x8D	此类型密钥对象实现了SM2PublicKey接口
public static final byte	TYPE_SM2_PRIVATE = 0x8E	此类型密钥对象实现了SM2PrivateKey接口
public static final short	LENGTH_SM4	SM4算法密钥长度值 LENGTH_SM4 = 128
public static final short	LENGTH_SM2_FP_256	SM2算法密钥长度值LENGTH_SM2_FP_256 = 256
protected	GMKeyBuilder ()	受保护的构造方法
public static Key	buildKey(byte keyType, short keyLength, boolean keyEncryption)	创建一个未初始化的密钥对象，用于签名和加解密运算

7.5.8.2 GMKeyBuilder 构造方法

7.5.8.2.1 声明

protected GMKeyBuilder();

7.5.8.2.2 描述

GMKeyBuilder类的构造器。

7.5.8.2.3 返回值

返回GMKeyBuilder类的实例。

7.5.8.3 buildKey

7.5.8.3.1 声明

```
public static key buildKey(
  byte keyType,
  short keyLength,
  boolean keyEncryption
) throws CryptoException;
```

7.5.8.3.2 描述

用于创建密钥对象的实例，内部封装了各种类型的密钥。

7.5.8.3.3 参数

keyType　　- 指定密钥的类型，定义为GMKeyBuilder.TYPE_ ..的常量。
keyLength　　- 指定密钥的长度，定义为GMKeyBuilder.LENGTH_ ..的常量。
keyEncryption　- 指定密钥是否需要加密。

7.5.8.3.4 返回值

返回Key对象。

7.5.8.3.5 抛出异常

CryptoException.NO_SUCH_ALGORITHM - 如果载体不支持的密钥类型抛出该异常。

7.5.9 GMCipher

7.5.9.1 概述

GMCipher类是加密算法的抽象基类。实现密码算法必须继承该类并实现所有抽象方法。

“填充”是指使用下文提到的指定方式扩充待加密数据块的内容，以达到算法要求的长度。在密码算法中，填充后数据一般为16的倍数。

非对称密钥算法使用公钥（加密）或私钥（签名）进行加密。另外，它们使用私钥（解密）或公钥（验证）解密。

GMCipher类的对象在拔卡或复位事件发生后重置到调用init()方法时的状态。当GMCipher类的对象对应的密钥失效时（遇到密钥对象相关的清除（clear）事件时），GMCipher类的对象也恢复到未初始化状态。

对于GMCipher类实例计算的中间结果，不需要进行事务管理。

GMCipher类定义见表11。

表11 GMCipher

类型	定义	说明
public static final byte	ALG_SM4_CBC_NOPAD = 0x89	加密算法使用SM4算法CBC模式，不进行填充
public static final byte	ALG_SM4_CBC_ISO9797_M1 = 0x8A	加密算法使用SM4算法CBC模式，采用ISO9797方法1填充方式

类型	定义	说明
public static final byte	ALG_SM4_CBC_ISO9797_M2 = 0x8B	加密算法使用SM4算法CBC模式，采用ISO9797方法2填充方式
public static final byte	ALG_SM4_CBC_PKCS5 = 0x8C	加密算法使用SM4算法CBC模式，采用PKCS5填充方式
public static final byte	ALG_SM4_ECB_NOPAD = 0x8D	加密算法使用SM4算法ECB模式，不进行填充
public static final byte	ALG_SM4_ECB_ISO9797_M1 = 0x8E	加密算法使用SM4算法ECB模式，采用ISO9797方法1填充方式
public static final byte	ALG_SM4_ECB_ISO9797_M2 = 0x8F	加密算法使用SM4算法ECB模式，采用ISO9797方法2填充方式
public static final byte	ALG_SM4_ECB_PKCS5 = 0x90	加密算法使用SM4算法ECB模式，采用PKCS5填充方式
public static final byte	ALG_SM2_WITH_SM3_NOPAD = 0xA1	加密算法使用SM3作为摘要算法，使用SM2进行加解密，不对数据进行填充
protected	GMCipher()	受保护的构造方法
public static final Cipher	getInstance (byte algorithm_id, boolean externalAccess)	根据指定的算法，创建对应的Cipher类的实例

注：1.getInstance()方法获取SM系列算法的加密对象实例；该对象实例实现了标准Cipher类的接口。

7.5.9.2 GMCipher 构造方法

7.5.9.2.1 声明

```
protected GMCipher();
```

7.5.9.2.2 描述

GMCipher类的构造器。

7.5.9.2.3 返回值

返回GMCipher类的实例。

7.5.9.3 getInstance

7.5.9.3.1 声明

```
public static final Cipher getInstance(
  byte algorithm_id,
  boolean externalAccess
) throws CryptoException;
```

7.5.9.3.2 描述

创建一个选定算法的Cipher对象实例。

7.5.9.3.3 参数

algorithm_id　　- 指定所创建对象的算法。

externalAccess　- 为true时代表所创建对象可以在多个应用实例中共享，并可以在所属应用实例不是当前选中实例时被访问。此时不可以为对象中的临时数据指定CLEAR_ON_DESELECT属性的空间。

7.5.9.3.4　返回值

返回指定算法的GMCipher类对象。

7.5.9.3.5　抛出异常

CryptoException.NO_SUCH_ALGORITHM - 当指定的算法不支持时，抛出该异常。

7.5.10　GMSM2KeyExchange

7.5.10.1　概述

GMSM2KeyExchange类是密码算法中SM2密钥交换算法类。实现密码算法中SM2密钥交换算法相关内容。

GMSM2KeyExchange类定义见表12。

表12　GMSM2KeyExchange

类型	定义	说明
public static final byte	SEND_MODE = 1	密钥交换过程为发起方
public static final byte	RECIVE_MODE = 0	密钥交换过程为响应方
public static final byte	KEY_LEN_8 = 8	交换8字节长密钥
public static final byte	KEY_LEN_16 = 16	交换16字节长密钥
public static final byte	PARAM_THIS_ID = 1	设置参数为己方ID
public static final byte	PARAM_OTHER_ID = 2	设置参数为对方ID
public static final byte	PARAM_THIS_FIX_PUBLICKEY = 3	设置参数为己方固定公钥
public static final byte	PARAM_THIS_FIX_PRIVATEKEY = 4	设置参数为己方固定私钥
public static final byte	PARAM_THIS_TEMP_PUBLICKEY_X = 7	设置参数为己方临时公钥X部分
public static final byte	PARAM_THIS_TEMP_PRIVATEKEY = 5	设置参数为己方临时私钥
public static final byte	PARAM_OTHER_FIX_PUBLICKEY = 6	设置参数为对方固定公钥
public static final	PARAM_OTHER_TEMP_PUBLICKEY = 8	设置参数为对方临时公钥

类型	定义	说明
byte		
protected	GMSM2KeyExchange()	无参数的构造函数，用于创建对象
public static GMSM2KeyExchange	getInstance()	创建GMSM2KeyExchange实例
public void	resetParam()	清除设置参数和初始化状态
public boolean	isInitialized()	判断GMSM2KeyExchange对象是否已被初始化
public void	setParam(byte[] buffer, short offset, short length, byte type)	设置SM2密钥交换所需参数
public short	getParam(byte[] buffer, short offset, byte type)	获取已写入的参数
public void	SM2KeyExchange(byte[] buffer, short offset, byte length, byte type)	使用密钥值进行SM2密钥交换方法

7.5.10.2 GMSM2KeyExchange

7.5.10.2.1 声明

protected GMSM2KeyExchange();

7.5.10.2.2 描述

构造函数，生成密钥交换对象。

7.5.10.3 getInstance

7.5.10.3.1 声明

Public static GMSM2KeyExchange getInstance ();

7.5.10.3.2 描述

创建GMSM2KeyExchange实例。

7.5.10.3.3 返回值

返回GMSM2KeyExchange类对象。

7.5.10.4 resetParam

7.5.10.4.1 声明

Public void resetParam();

7.5.10.4.2 描述

清除已设置参数和初始化状态。

7.5.10.5 isInitialized

7.5.10.5.1 声明

Public boolean isInitialized ();

7.5.10.5.2 描述

判断GMSM2KeyExchange对象是否已被初始化。

7.5.10.5.3 返回值

True — GMSM2KeyExchange对象已被初始化。
False - GMSM2KeyExchange对象未被初始化。

7.5.10.6 setParam

7.5.10.6.1 声明

Public void setParam(byte[] buffer, short offset, short length, byte type);

7.5.10.6.2 描述

以数组形式设置密钥交换所用各项参数。

7.5.10.6.3 参数

buffer - 参数所在数组。
offset - 参数偏移。
length - 参数长度。
type - 参数类型，如PARAM_THIS_ID表明设置的参数为己方ID。

7.5.10.6.4 抛出异常

NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上参数长度超出buffer数据长度时，抛出该异常。
CryptoException.ILLEGAL_VALUE - 输入参数有误抛出该异常。

7.5.10.7 getParam

7.5.10.7.1 声明

Public short getParam(byte[] buffer, short offset, byte type);

7.5.10.7.2 描述

获取已经写入的密钥交换所用的参数。

7.5.10.7.3 参数

buffer - 读出数据所在数组。
offset - 读出数据位置偏移。
type - 读出数据类型，如PARAM_THIS_ID表明读出己方ID参数。

7.5.10.7.4 返回值

读出数据长度。

7.5.10.7.5 抛出异常

NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上参数长度超出buffer数据长度时，抛出该异常。
CryptoException.UNINITIALIZED_KEY - 相应参数处于未初始化状态时抛出该异常。
CryptoException.ILLEGAL_VALUE - 输入参数有误抛出该异常。

7.5.10.8 SM2KeyExchange

7.5.10.8.1 声明

Public void SM2KeyExchange(byte[] buffer, short offset, byte length, byte type);

7.5.10.8.2 描述

执行密钥交换计算。

7.5.10.8.3 参数

buffer - 结果数据所在数组。
offset - 结果数据偏移。
length - 交换的密钥长度参数，如KEY_LEN_8表示交换8字节密钥。
type - 计算类型，如SEND_MODE表示发起方计算。

7.5.10.8.4 抛出异常

NullPointerException - buffer参数为null时抛出该异常。
ArrayIndexOutOfBoundsException - offset参数为负数或者超过buffer数组时，或者offset加上参数长度超出buffer数据长度时，抛出该异常。
CryptoException.ILLEGAL_VALUE - 输入参数有误抛出该异常。
CryptoException.UNINITIALIZED_KEY - 参数没有设置完全抛出该异常。

7.5.11 GMCipherExtend

7.5.11.1 概述

GMCipherExtend类是密码算法扩展类,用于实现在GMCipher和GMSM2KeyExchange类中没有涉及的密码算法相关计算方法。

GMCipherExtend类定义见表13。

表13 GMCipherExtend

类型	定义	说明
public static final byte	PARAM_FP_256	使用内置Fp-256参数计算Za值
public	GMCipherExtend()	无参数的构造函数，用于创建对象
public static short	getZa(byte[] IDBuffer, short IDOffset, short IDLength, byte[] PubKeyBuffer, short PubKeyOffset,	使用传入的公钥（x,y）计算Za方法

类型	定义	说明
	byte[] destBuffer, short destOffset, byte type)	
public static short	getZa(byte[] IDBuffer, short IDOffset, short IDLength, SM2PublicKey PubKey, byte[] destBuffer, short destOffset, byte type)	使用传入的公钥对象计算Za方法
public static short	generateSM2Pubkey (SM2PublicKey public_key, SM2PrivateKey private_key)	根据已知的私钥对象，输出对应的公钥信息。
public static short	generateSM2Pubkey (byte[] pubKey, short offset, SM2PrivateKey private_key)	根据已知的私钥对象，输出对应的公钥信息。

7.5.11.2 GMCipherExtend

7.5.11.2.1 声明

Public GMCipherExtend();

7.5.11.2.2 描述

GMCipherExtend构造函数，用于创建GMCipherExtend对象。

7.5.11.3 getZa

7.5.11.3.1 声明

Public short getZa(byte[] IDBuffer, short IDOffset, short IDLength, byte[]PubKeyBuffer, short PubKeyOffset, byte[] destBuffer, short destOffset, byte type);

7.5.11.3.2 描述

使用传入的ID和公钥值产生Za值。

7.5.11.3.3 参数

IDBuffer - ID值所在数组。
IDOffset - ID值偏移。
IDLength - ID值长度。
PubKeyBuffer - 公钥值所在数组。
PubKeyOffset - 公钥值偏移。
destBuffer - Za值存放数组。
destBuffer - Za值偏移。
type - 曲线参数类型，目前仅支持PARAM_FP_256一种模式。

7.5.11.3.4 返回值

Za长度。

7.5.11.3.5 抛出异常

CryptoException.ILLEGAL_VALUE - 输入参数有误时，抛出该异常。

7.5.11.4 getZa

7.5.11.4.1 声明

Public short getZa(byte[] IDBuffer, short IDOffset, short IDLength, SM2PublicKey PubKey, byte[] destBuffer, short destOffset, byte type);

7.5.11.4.2 描述

使用传入的ID和公钥对象计算Za值。

7.5.11.4.3 参数

IDBuffer - ID值所在数组。
IDOffset - ID值偏移。
IDLength - ID值长度。
PubKey - 公钥对象。
destBuffer - Za值存放数组。
destOffset - Za值偏移。
type - 曲线参数类型，目前仅支持PARAM_FP_256一种模式。

7.5.11.4.4 返回值

Za长度。

7.5.11.4.5 抛出异常

CryptoException.ILLEGAL_VALUE - 输入参数有误时，抛出该异常。

7.5.11.5 generateSM2Pubkey

7.5.11.5.1 声明

public static short generateSM2Pubkey (SM2PublicKey public_key, SM2PrivateKey private_key)

7.5.11.5.2 描述

根据已知的私钥对象，输出对应的公钥信息。

7.5.11.5.3 参数

public_key - 公钥数据的输出对象。
private_key - 已知的私钥对象。

7.5.11.5.4 返回值

公钥长度。

7.5.11.5.5 抛出异常

CryptoException.INVALID_INIT - 私钥未初始化时，抛出该异常。
NullPointerException – 公钥对象为空时，抛出该异常。

7.5.11.6 generateSM2Pubkey

7.5.11.6.1 声明

public static short generateSM2Pubkey (byte[] pubKey, short offset, SM2PrivateKey private_key)

7.5.11.6.2 描述

根据已知的私钥对象，输出对应的公钥信息。

7.5.11.6.3 参数

pubKey - 公钥数据的输出数组。
offset - 公钥数据的输出偏移。
private_key - 已知的私钥对象。

7.5.11.6.4 返回值

公钥长度。

7.5.11.6.5 抛出异常

CryptoException.INVALID_INIT - 私钥未初始化，抛出该异常。
NullPointerException – 公钥数据输出数组为空，抛出该异常。

附 录 A
（资料性附录）
多应用安全管理的密码应用要求

A.1 简介

本部分描述了多应用载体中多应用安全管理的密码使用要求。多应用管理采用了安全域的技术方案，由每个安全域划分出一个独立的逻辑安全区，该逻辑安全区域可以配置独立的密码。安全域中密码使用环节包括：

——安全域中密码算法标识获取；

——载体外实体和安全域的安全通道管理；

——CAP包签名与校验；

——应用下载、安装、迁移、删除的授权令牌；

——委托管理收条。

载体多应用安全管理技术框架见图A.1。

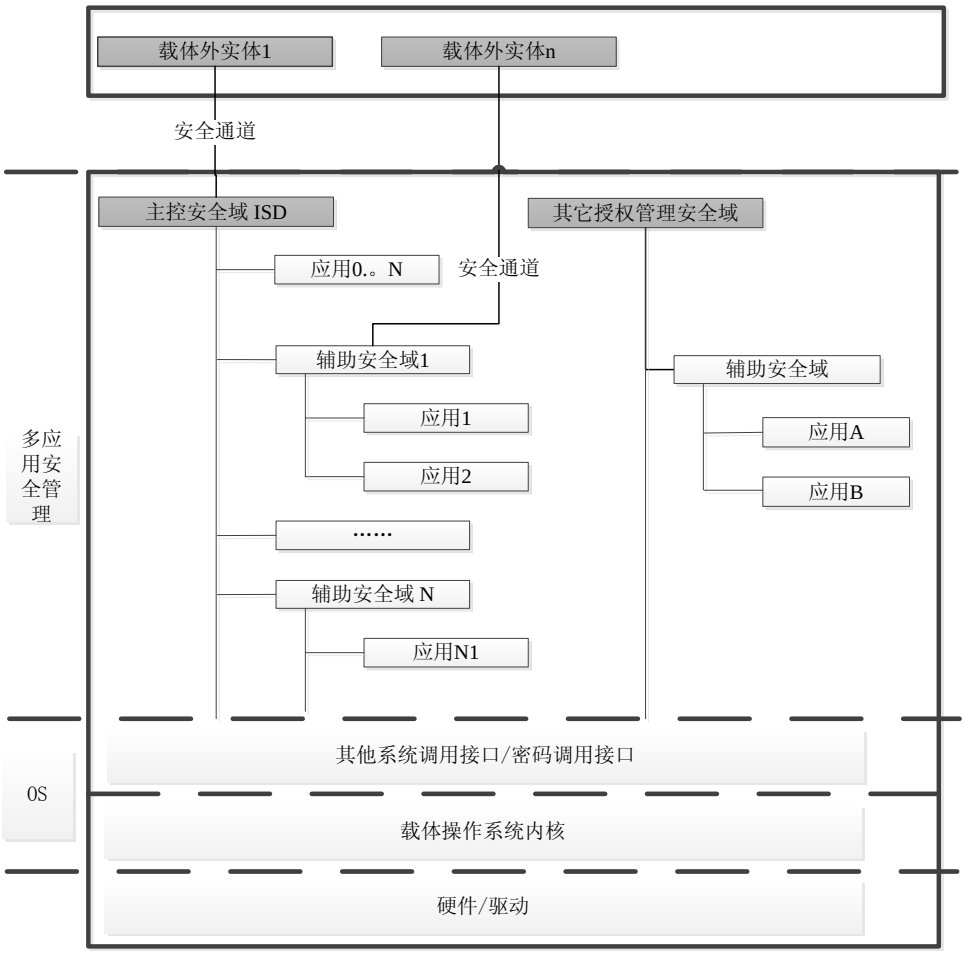

图 A.1 载体多应用安全管理技术框架

该框架中，多应用安全管理环境使用安全域这一技术机制，实现了对载体上多个应用的安全管理。每个安全域都可以有自己独立的密码算法、密钥以及安全策略。载体外实体通过安全域进行安全的应用访问和应用管理。其中主控安全域由称为发行方的载体外实体持有，提供各辅助安全域的生命周期管理和应用管理授权；辅助安全域由主控安全域创建，创建完成后可交由其他载体外实体持有；应用通过安全通道下载到主控或辅助安全域，并由具备权限的安全域进行管理授权。

A.2 密码算法标识获取

为了保证安全域中密码算法的独立可配置特性，在安全域中定义一个密码算法标签(以下简称标签(Tag))，标识该安全域中安全通道、应用管理授权、委托管理收条、包签名和校验的算法及其参数。载体外实体可以通过读取该标签获得安全域上的算法能力；在应用管理中，载体上的基础服务功能也可以通过该标签获取相应的算法能力。

Tag 标签定义见表 A.1，数值为 ：‘5A’。

表 A.1 Tag 标签定义

Tag	长度	含义
‘0x00,0x70’	1-n	数据组信息
‘0x5A’	3-20	密码算法标识数据元

密码算法标识数据元定义见表 A.2。

表 A.2 密码算法标识数据元

T	L	V	是否可选
0xA5	3	SCP02 的算法标识：缺省值‘0x19’‘0x19’‘0x18’	可选
0xA6	2	SCP11 的杂凑算法标识：‘0x20’ SCP11 的签名算法标识：缺省值‘0x28’	可选
0xA7	2	DAP 验证的杂凑算法标识：缺省值‘0x20’ DAP 验证的签名算法标识：缺省值‘0x28’	可选
0xA8	2	应用管理授权令牌算法标识：缺省值‘0x20’ 应用管理授权令牌算法标识：缺省值‘0x28’	可选
0xA9	1	委托管理收条算法标识：缺省值‘0x19’	可选

安全域的密码算法标签 Tag 应在个人化阶段赋值，个人化完成后，在载体生命周期内不允许修改。

A.3 安全通道的密码应用规则

A.3.1 SCP02安全通道

SCP02安全通道应能够支持本规范中SM4对称密码算法。 载体外实体可以先通过getdata获得SCP02的密码算法标识来确认使用的算法，或通过缺省约定来确认使用的算法。

当选择SM4算法时，安全域的加密、MAC、敏感数据三个初始化密钥由主控安全域通过PUT KEY命令写入到安全域中， PUT KEY命令的Check value取左三个字节。当载体外实体打开SCP02安全通道时，会话密钥的生成规则符合SCP02安全通道的要求。

当选择SM4算法时，敏感数据加密使用ECB(128位)模式，加密和MAC使用CBC(128位)模式。加密和MAC计算的数据元具体的运算过程如下：

a）过程密钥计算

按照 SCP02 的计算过程计算，使用 SM4 算法。

b）卡密文计算

卡密文（Card cryptogram） 实现需要连接卡外生成的 8 字节 Host Challenge 和 2 字节的序列计数器以及卡内生成的 6 个字节 Card Challenge 产生 16 个字节的数据块，不填充。

签名方法采用 S-ENC 会话密钥和全部为二进制 0 的 ICV 作用于这 16 字节数据块，SM4 加密后取后 8 个 byte 就是卡的鉴别密码。

c）主机密文

主机密文（host cryptogram） 的实现需要连接 2 字节的序列计数器、以及卡内生成的 6 字节 Card Challenge 和卡外生成的 8 字节 Host Challenge 产生 16 个字节的数据块。

签名方法采用 S-ENC 会话密钥和全部为二进制 0 的 ICV 作用于这 16 字节数据块，SM4 加密后取前 8 个 byte 就是主机的鉴别密码。

d）MAC 计算

按照 SCP02 的计算过程，算法为 SM4，得到的计算结果取前 8 个 byte 作为 MAC。下次的 ICV 使用上一次的 MAC(如果是 i=15 需先加密)+8byte’00’组合成 16byte 数据。

A.3.2 SCP11安全通道

SCP11安全通道应能够支持本规范中的SM2算法。载体外实体可以先通过getdata获得SCP11的密码算法标识来确认使用的算法，或通过缺省约定来确认使用的算法。

会话密钥生成遵循SCP11流程，使用SM4算法，其中密钥交换协议遵循GB/T 32918的第3部分，使用密钥交换结果作为过程密钥。

SM4作为会话密钥，后续使用方式如加解密、填充、签名规则遵循SCP03要求。

SCP11所涉及到证书格式参考附录B。

A.4 CAP包签名与校验

CAP的规格遵循标准包规格。

在应用CAP下载过程中，LOAD命令的第一条中应包含供安全域验证的DAP块，载体上多应用安全管理环境应通过安全域AID获取安全域的密码算法标识，并使用该算法进行校验。

CAP 包的格式见表 A.3。

表 A.3 CAP 包格式

Tag	长度	含义
‘0xE2’	1-n	DAP 块
‘0x4F’	5-16	安全域 AID
‘0xC3’	1-n	下载文件数据块签名
‘0xC4’	1-n	下载文件数据块开始

A.5 应用管理授权令牌

A.5.1 下载令牌

下载令牌（Load Token）是允许应用程序加载到载体上的令牌，用以验证 Load 请求的合法性。Load Token 计算结构见图 A.2。

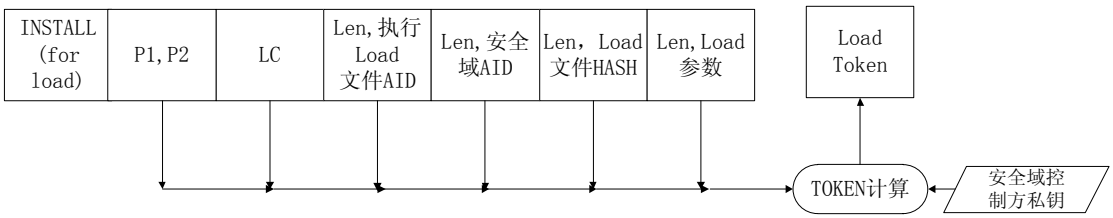

图 A.2 Load Token 计算输入结构

根据安全域的密码算法配置：

——Load Token 的输入字段中， Load 文件的 Hash 值长度为 32 字节；

——Load Token 的值长度为 64 字节。

Load Token 的输入字段定义见表 A.4。

表 A.4 Load Token 输入字段

字段	长度
P1	1
P2	1
LC	1
Load 文件 AID 长度	1
Load 文件 AID	5-16
安全域 AID 长度	1
安全域 AID	5-16
Load 文件 HASH 值长度	1
Load 文件 HASH 值	32
Load 参数域长度	1-2
Load 参数域	0-n

A.5.2 安装令牌

安装令牌（Install Token）是一个包含 Load 文件的应用签名，用于验证载体中应用的安装是否被合法授权。当使用密码算法时， Install Token 的值为 64 字节。

A.5.3 迁移令牌

迁移令牌（Extradition Token）是授权应用从一个安全域迁移到另一个安全域的令牌。当使用密码算法时， Extradition Token 的值为 64 字节。

A.5.4 删除令牌

删除令牌(Delete Token)是授权应用从安全域中删除的令牌。当使用商用密码算法时，Delete Token 的值为 64 字节。

A.6 委托管理收条

委托管理收条用于应用下载、安装、删除时，载体上安全域向载体外实体提供的操作结果凭证。当使用密码算法时， 委托管理收条的值为 16 字节。

附 录 B
多应用安全管理的证书格式

B.1 多应用安全管理证书格式

本证书应用于多应用安全管理，使用SM2算法，格式符合X.509规范。具体要求如表B.1所示：

表 B.1 多应用管理证书格式

TAG	名称					长度	值
0x30	certificate					0x01~0x11B	
	0x30	tbsCertificate				0x01~0xC1	
		0xA0	version			0x03	
			0x02			0x01	0x02
		0x02	serialNumber			0x01~0x10	
		0x30	Signature Algorithm			0x0C	
			0x06	OID		0x08	2A 81 1C CF 55 01 83 75
			0x05	NULL		0x00	
		0x30	issuer			0x01~0x12	
			0x02	INTEGER		0x01~0x10	
		0x30	Expiration Date(BCD 格式)			0x06	
			0x04	OCTET STRING		0x04	
		0x30	Subject Identifier			0x01~0x12	
			0x02	INTEGER		0x01~0x10	
		0x30	Public key			0x59	
			0x30			0x13	
				0x06	OID	0x07	2A 86 48 CE 3D 02 01
				0x06	OID	0x08	2A 81 1C CF 55 01 82 2D
			0x03	BIT STRING		0x42	00 04 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 04 表示 SM2 密钥 后面 32 字节表示公钥 X 后面 32 字节表示公钥 Y
		0xA3	extend			0x0F	
			0x30			0x0D	
				0x30	Key usage	0x0B	
					0x06 OID	0x03	55 1D 0F

<table>
<tr><th>TAG</th><th colspan="8">名称</th><th>长度</th><th>值</th></tr>
<tr><td rowspan="11"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td>0x04</td><td colspan="2">OCTET STRING</td><td>0x04</td><td></td></tr>
<tr><td></td><td>0x03</td><td>BIT STRING</td><td>0x02</td><td>01 06</td></tr>
<tr><td rowspan="3">0x30</td><td colspan="5">Signature Algorithm</td><td>0x0C</td><td></td></tr>
<tr><td>0x06</td><td colspan="4">OID</td><td>0x08</td><td>2A 81 1C CF 55 01 83 75</td></tr>
<tr><td>0x05</td><td colspan="4">NULL</td><td>0x00</td><td></td></tr>
<tr><td rowspan="4">0x03</td><td colspan="5">Signature value</td><td>0x48</td><td></td></tr>
<tr><td>0x30</td><td colspan="4"></td><td>0x45</td><td></td></tr>
<tr><td rowspan="2"></td><td>0x02</td><td colspan="3">INTEGER</td><td>0x21</td><td>签名后的数据 R</td></tr>
<tr><td>0x02</td><td colspan="3">INTEGER</td><td>0x20</td><td>签名后的数据 S</td></tr>
</table>

参 考 文 献

[1]GB/T 15852.1-2008 信息技术 安全技术 消息鉴别码
[2]GM/T 0009-2012 SM2密码算法使用规范
[3]GM/T 0015-2012 基于SM2密码算法的数字证书格式规范
[4]GM/T 0044-2016 SM9标识密码算法
[5]GM/Z 4001-2013 密码术语