ICS 33.050
CCS M30

团　体　标　准

T/TAF 333—2026

基于SIM卡的GBA_U派生密钥API技术要求

Technical requirements for API based on keys derived from GBA_U of SIM card

2026-02-09 发布　　2026-02-09 实施

电信终端产业协会　发布

版权声明

邮箱：tafrb@taf.org.cn

电话：010-82052809

目　　次

前　言

本文件按照 GB/T 1.1—2020《标准化工作导则 第 1 部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由电信终端产业协会（TAF）提出并归口。

本文件起草单位：中国移动通信有限公司研究院、中移物联网有限公司、芯昇科技有限公司、北京华弘集成电路设计有限责任公司、中国信息通信研究院。

本文件主要起草人：霍薇靖、李一萌、刘梅娟、王晓珍、王雷、袁琦、张宏星。

基于SIM卡的GBA_U派生密钥API技术要求

1 范围

本文件规定了基于 SIM 卡的 GBA_U 派生密钥 API 技术要求，定义基于 SIM 卡的 GBA_U 派生密钥 Ks_int_NAF 的 API 接口，SIM 卡上其他应用可调用该接口进行加解密及签名验签操作，本文件还定义了 SIM 卡上应用调用 GBA_U API 的访问控制规则。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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。

3GPP TS 31.102 USIM应用特性(Characteristics of the USIM Application)

3GPP TS 31.103 IP多媒体业务识别模块（ISIM）应用特性(Characteristics of the IP Multimedia Services Identity Module (ISIM) application)

3GPP TS 33.220 通用认证架构（GAA）；通用引导(Generic Authentication Architecture (GAA);Generic Bootstrapping)

3 术语和定义

本文件没有需要界定的术语和定义。

4 缩略语

下列缩略语适用于本文件。

ADF：应用专用文件（Application Dedicated File）

ADM：在创建EF的管理者控制下的对此EF的存取条件（Access condition to an EF which is under the control of the authority which creates this file）

AID：应用标识符（Application Identifier）

API：应用程序接口（Application Programming Interface）

BSF：引导服务器功能（Bootstrapping Server Function）

COS：用户卡操作系统（Chip Operation System）

GBA：通用引导架构（Generic Bootstrapping Architecture）

GBA_U：基于UICC的GBA（Generic Bootstrapping Architecture UICC）

ISIM：IP多媒体服务身份模块（IM Services Identity Module）

NAF：网络应用功能（Network Application Function）

SIM：移动网络用户身份模块（Subscriber Identifier Module）

UICC：通用集成电路卡（Universal Integrated Circuit Card）

USIM：通用用户识别模块（Universal Subscriber Identity Module）

5 概述

GBA_U API在SIM卡中的位置如图1所示，位于SIM卡COS层的USIM API中，为基于GBA_U鉴权能力派生的密钥Ks_int_NAF为基础定义的密码服务接口，可为SIM卡上的应用（即applet）提供数据加解密、数字签名及验证等各类密码运算服务，密钥派生过程参见附录A，其package对应的AID见附录B。

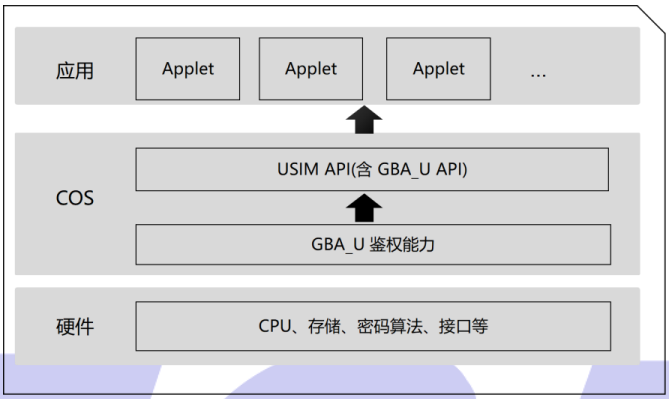

图 1 GBA_U API 在 SIM 卡中的位置

6 GBA_U API

6.1 GBAUCipher

6.1.1 概述

GBAUCipher 提供基于 Ks_int_NAF 的加解密功能，方法定义见表 1。

表1 GBAUCipher方法定义

方法	说明
getInstance(byte algorithm, boolean externalAccess)	创建 GBAUCipher类的实例
getInstance(byte cipherAlgorithm, byte paddingAlgorithm, boolean externalAccess)	
init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen)	使用NAF ID以及对应的Ks_int_NAF密钥初始化GBAUCipher对象
init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, byte[] bArray, short bOff, short bLen)	
init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, short keyLength)	
init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, byte[] bArray, short bOff, short bLen, short keyLength)	

表1 GBAUCipher方法定义（续）

方法	说明
update(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)	输入更多数据，进行多部分的加密/解密
doFinal(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)	使用init方法指定的NAF ID及关联的Ks_int_NAF密钥，对所有输入数据执行加密/解密操作
getAlgorithm()	获取加密算法
getCipherAlgorithm()	获取原始加密算法
getPaddingAlgorithm()	获取填充算法

使用GBAUCipher类进行加密/解密操作的流程如下：

首先使用getInstance()方法创建实例，再调用init()方法进行初始化操作，指定NAF ID以及对应的Ks_int_NAF密钥，如果涉及数据量较大数据的加密/解密，则调用update()方法，最后调用doFinal()方法完成最终的加密/解密操作。

6.1.2 getInstance 方法 1

6.1.2.1 声明

创建GBAUCipher类的实例。

public static GBAUCipher getInstance(byte algorithm, boolean externalAccess)

6.1.2.2 参数

algorithm：算法。

externalAccess：若为true，表示该实例可在多个卡应用实例间共享，且当GBAUCipher实例的所有者不是当前选中的卡应用时，也可通过Shareable接口访问该实例。若为false，实现不得为内部数据分配CLEAR_ON_ AESELECT类型的瞬态空间。

6.1.2.3 返回

对应算法的GBAUCipher对象。

6.1.2.4 异常抛出

若请求的算法不支持，则抛出CryptoException.NO_SUCH_ALGORITHM。

6.1.3 getInstance 方法 2

6.1.3.1 声明

创建 GBAUCipher 类的实例。

public static GBAUCipher getInstance(byte cipherAlgorithm, byte paddingAlgorithm, boolean externalAccess)

6.1.3.2 参数

cipherAlgorithm：加密算法，例如CIPHER_AES_CBC。

paddingAlgorithm：填充算法，例如PAD_NULL。

externalAccess：若为true，表示该实例可在多个卡应用实例间共享，且当 GBAUCipher 实例的所有者不是当前选中的卡应用时，也可通过Shareable接口访问该实例。若为false，实现不得为内部数据分配CLEAR_ON_ AESELECT类型的瞬态空间。

6.1.3.3 返回

对应算法的GBAUCipher对象。

6.1.3.4 异常抛出

若请求的算法不支持，则抛出CryptoException.NO_SUCH_ALGORITHM。

6.1.4 init 方法 1

6.1.4.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥初始化GBAUCipher对象，该方法适用于不需要初始化参数或使用默认参数值的算法。

public abstract void init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen)

密钥长度约定：

——AES-128：使用256位Ks_int_NAF密钥的左侧128个最高有效位；

——AES-192：使用256位Ks_int_NAF密钥的左侧192个最高有效位；

——Korean SEED：使用256位Ks_int_NAF密钥的左侧128个最高有效位；

——SM4：使用256位Ks_int_NAF密钥的左侧128个最高有效位。

若使用此方法初始化CBC模式的AES、Korean SEED 或SM4算法，初始向量（IV）将默认设为0。

为优化性能，当Ks_int_NAF密钥为瞬态密钥时，实现应尽可能使用瞬态空间进行内部存储。

6.1.4.2 参数

theMode：解密模式或加密模式。

adfAID：包含ADF的AID值的字节数组（例如USIM或ISIM的AID）。

adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。

adfAIDLen：ADF AID 值的字节长度。

nafID：包含NAF ID值的字节数组。

nafIDOff：nafID数组中NAF ID值的起始偏移量。

nafIDLen：NAF ID值的字节长度。

6.1.4.3 返回

无。

6.1.4.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；

——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API；

——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义；
——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。
出现下述异常情况时，抛出javacard.security.CryptoException，具体原因码如下：
——CryptoException.ILLEGAL_USE：若theMode参数为未定义值；
——NullPointerException：若adfAID或nafID为空；
——ArrayIndexOutOfBoundsException：若对adfAID或nafID的检查操作导致访问数组越界。

6.1.5 init 方法 2

6.1.5.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥和特定算法初始化GBAUCipher对象。
public abstract void init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, byte[] bArray, short bOff, short bLen)
算法密钥使用规则：
——AES-128：使用256位Ks_int_NAF密钥的左侧128位最高有效位；
——AES-192：使用256位Ks_int_NAF密钥的左侧192位最高有效位；
——Korean SEED：使用256位Ks_int_NAF密钥的左侧128位最高有效位；
——SM4：使用256位Ks_int_NAF密钥的左侧128位最高有效位。
CBC模式初始向量(IV)要求：
——AES（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)；
——KoreanSEED（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)；
——SM4（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)。
ECB模式限制：
AES（ECB模式）、Korean SEED（ECB模式）、SM4（ECB模式）：调用时抛出CryptoException.ILLEGAL_VALUE异常。
性能优化建议：
当Ks_int_NAF密钥为瞬态密钥时，实现应尽可能使用瞬态空间进行内部存储，以提升性能。

6.1.5.2 参数

theMode：解密模式或加密模式。
adfAID：包含ADF AID值的字节数组（例如USIM或ISIM的AID）。
adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。
adfAIDLen：ADF AID 值的字节长度。
nafID：包含NAF ID值的字节数组。
nafIDOff：nafID数组中NAF ID值的起始偏移量。
nafIDLen：NAF ID值的字节长度。
bArray：包含特定算法初始化信息的字节数组。
bOff：bArray数组中特定算法的数据的起始偏移量。
bLen：特定算法的参数数据的字节长度。

6.1.5.3 返回

无。

6.1.5.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；

——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API；

——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义；

——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，具体原因码如下：

——CryptoException.ILLEGAL_USE：若theMode参数为未定义值；

——NullPointerException：若adfAID或nafID或bArray为空；

——ArrayIndexOutOfBoundsException：若对adfAID或nafID或bArray的检查操作导致访问数组越界。

6.1.6 init 方法 3

6.1.6.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥初始化GBAUCipher对象，该方法适用于不需要初始化参数或使用默认参数值的算法。

public abstract void init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, short keyLength)

密钥长度约定：

——AES-128：使用256位Ks_int_NAF密钥的左侧128个最高有效位；

——AES-192：使用256位Ks_int_NAF密钥的左侧192个最高有效位；

——Korean SEED：使用256位Ks_int_NAF密钥的左侧128个最高有效位；

——SM4：使用256位Ks_int_NAF密钥的左侧128个最高有效位。

若使用此方法初始化CBC模式的AES、Korean SEED 或SM4算法，初始向量（IV）将默认设为 0。

为优化性能，当Ks_int_NAF 密钥为瞬态密钥时，实现应尽可能使用瞬态空间进行内部存储。

6.1.6.2 参数

theMode：解密模式或加密模式。

adfAID：包含ADF的AID值的字节数组（例如USIM或ISIM的AID）。

adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。

adfAIDLen：ADF AID 值的字节长度。

nafID：包含NAF ID值的字节数组。

nafIDOff：nafID数组中NAF ID值的起始偏移量。

nafIDLen：NAF ID值的字节长度。

keyLength：密钥长度 - 密钥的位数。有效的密钥位数长度取决于密钥类型。

6.1.6.3 返回

无。

6.1.6.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；
——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API；
——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义；
——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。
出现下述异常情况时，抛出javacard.security.CryptoException，具体原因码如下：
——CryptoException.ILLEGAL_USE:若theMode参数为未定义值,或keyLength参数与getInstance()传入算法不适配；
——NullPointerException：若adfAID或nafID为空；
——ArrayIndexOutOfBoundsException：若对adfAID或nafID的检查操作导致访问数组越界。

6.1.7 init 方法 4

6.1.7.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥和特定算法初始化GBAUCipher对象。

public abstract void init(byte theMode, byte[] adfAID, short adfAIDOff, short adfAIDLen, byte[] nafID, short nafIDOff, short nafIDLen, byte[] bArray, short bOff, short bLen, short keyLength)

算法密钥使用规则：
——AES-128：使用256位Ks_int_NAF密钥的左侧128位最高有效位；
——AES-192：使用256位Ks_int_NAF密钥的左侧192位最高有效位；
——Korean SEED：使用256位Ks_int_NAF密钥的左侧128位最高有效位；
——SM4：使用256位Ks_int_NAF密钥的左侧128位最高有效位。
CBC模式初始向量(IV)要求：
——AES（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)；
——KoreanSEED（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)；
——SM4（CBC模式）：需要在bArray参数中提供16字节的初始向量(IV)。
ECB模式限制：
AES（ECB模式）、Korean SEED（ECB模式）、SM4（ECB模式）：调用时抛出CryptoException.ILLEGAL_VALUE异常。
性能优化建议：
当Ks_int_NAF密钥为瞬态密钥时，实现应尽可能使用瞬态空间进行内部存储，以提升性能。

6.1.7.2 参数

theMode：解密模式或加密模式。
adfAID：包含ADF AID值的字节数组（例如USIM或ISIM的AID）。
adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。
adfAIDLen：ADF AID 值的字节长度。
nafID：包含NAF ID值的字节数组。
nafIDOff：nafID数组中NAF ID值的起始偏移量。
nafIDLen：NAF ID值的字节长度。
bArray：包含特定算法初始化信息的字节数组。
bOff：bArray数组中特定算法的数据的起始偏移量。
bLen：特定算法的参数数据的字节长度。

keyLength：密钥长度 - 密钥的位数。有效的密钥位数长度取决于密钥类型。

6.1.7.3 返回

无。

6.1.7.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；

——GBA_U_UNALLOWED_ACCESS：卡应用无权使用API（访问条件详见第7章节）；

——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义（访问条件详见第7章节）；

——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，具体原因码如下：

——CryptoException.ILLEGAL_USE:若theMode参数为未定义值,或keyLength参数与getInstance()传入算法不适配;

——NullPointerException：若adfAID或nafID或bArray为空;

——ArrayIndexOutOfBoundsException：若对adfAID或nafID或bArray的检查操作导致访问数组越界。

6.1.8 update 方法

6.1.8.1 声明

输入更多数据，使用init方法指定的与 NAF ID及关联的Ks_int_NAF密钥进行处理。

public abstract short update(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)

该方法需要临时存储中间结果。如果输入数据长度不是块大小的整数倍，需要分配额外的内部存储空间来存储部分输入数据块。

仅当密码算法所需的所有输入数据不能一次性提供在一个字节数组中时，才应使用此方法。

若所有输入数据位于单个字节数组中，建议使用 doFinal() 方法一次性处理全部数据。

必须调用 doFinal方法来完成对通过一次或多次调用 update方法缓冲的剩余输入数据处理。

若输入缓冲区（inBuff）和输出缓冲区（outBuff）为同一数组，输出区域不得部分重叠输入区域。例如，若 inBuff == outBuff 且 inOffset < outOffset < inOffset+inLength，可能导致输出错误。

输入数据未按块对齐时，不允许输入与输出缓冲区有任何重叠。例如，若 inBuff == outBuff 且 outOffset < inOffset+inLength，可能导致输出错误。

解密操作（ISO 9797 方法1填充除外）：填充字节不会写入outBuff。

加密/解密操作：块对齐要求可能导致输出字节数大于、小于输入长度（inLength），甚至为0。

若inLength为0，此方法不执行任何操作。

6.1.8.2 参数

inBuff：待加密/解密的数据输入缓冲区。

inOffset：输入缓冲区中开始加密/解密的起始偏移量。

inLength：待加密/解密的字节长度。

outBuff：输出缓冲区，可与输入缓冲区为同一数组。
outOffset：输出缓冲区中密文/明文结果的起始偏移量。

6.1.8.3 返回

返回写入 outBuff 的字节数。

6.1.8.4 异常抛出说明

javacard.security.CryptoException的原因码如下：
——CryptoException.UNINITIALIZED_KEY：密钥（Ks_int_NAF）未初始化；
——CryptoException.INVALID_INIT：GBAUCipher 对象未正确初始化；
——CryptoException.ILLEGAL_USE：输入消息长度不被支持，或消息值大于等于模数；
——NullPointerException：若inBuff或outBuff为空；
——ArrayIndexOutOfBoundsException：若对inBuff 或 outBuff的检查导致数组越界访问。

6.1.9 doFinal 方法

6.1.9.1 声明

使用init方法指定的NAF ID及关联的Ks_int_NAF密钥，对所有输入数据执行加密/解密操作。
public abstract short doFinal(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)
调用该方法后，GBAUCipher 对象会被重置为上一次通过 init() 方法初始化时的状态，可立即用于加密或解密更多数据。AES、 Korean SEED 和SM4算法使用的初始向量（IV）将被重置为0。
若输入缓冲区inBuff与输出缓冲区outBuff为同一数组，输出数据区域不得与输入数据区域部分重叠（即输入数据在使用前不能被修改）。例如：若inBuff == outBuff且inOffset < outOffset < inOffset+inLength，可能导致输出结果错误。
输入数据未按块对齐时，输入与输出缓冲区不允许任何重叠。例如：若inBuff == outBuff且outOffset < inOffset+inLength，可能导致输出结果错误。
算法状态与初始向量（IV）说明如下:
——CBC 模式算法（AES、Korean SEED、SM4）：doFinal()方法会将初始向量（IV）重置为 0，可通过init()方法重新初始化 IV；
——解密操作（ISO 9797方法1填充除外）：填充字节不会写入outBuff；
——输入/输出字节数差异：加密或解密操作中，写入outBuff的字节数可能大于、小于inLength，甚至为 0；
——解密操作若抛出ArrayIndexOutOfBoundsException，outBuff可能已被部分修改。

6.1.9.2 参数

inBuff：待加密/解密的数据输入缓冲区。
inOffset：输入缓冲区中开始加密/解密的起始偏移量。
inLength：待加密/解密的字节长度。
outBuff：输出缓冲区，可与输入缓冲区为同一数组。
outOffset：输出缓冲区中结果数据的起始偏移量。

6.1.9.3 返回

写入到输出缓冲区（outBuff）中的字节数。

6.1.9.4 异常抛出说明

javacard.security.CryptoException的原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若 GBAUCipher对象未初始化；

——CryptoException.ILLEGAL_USE：

若满足以下任一条件：

- 此 GBAUCipher 算法不填充消息且消息未按块对齐；
- 此 GBAUCipher 算法不填充消息且未通过inBuff或update方法提供输入数据；
- 解密数据未被适当的填充字节界定。

——NullPointerException：若inBuff或outBuff为空；

——ArrayIndexOutOfBoundsException：若对inBuff或outBuff的检查操作导致访问数组越界。

6.1.10 getAlgorithm 方法

6.1.10.1 声明

获取加密算法。

```
public abstract byte getAlgorithm()
```

6.1.10.2 参数

无。

6.1.10.3 返回

返回算法代码，如果该算法不属于预定义算法之一，则返回0。

6.1.10.4 异常抛出说明

无。

6.1.11 getCipherAlgorithm 方法

6.1.11.1 声明

获取原始加密算法。

```
public abstract byte getCipherAlgorithm()
```

6.1.11.2 参数

无。

6.1.11.3 返回

返回原始算法代码，如果该算法不属于预定义算法之一，则返回0。

6.1.11.4 异常抛出说明

无。

6.1.12 getPaddingAlgorithm 方法

6.1.12.1 声明

获取填充算法。

public abstract byte getPaddingAlgorithm()

6.1.12.2 参数

无。

6.1.12.3 返回

返回填充算法代码，如果该算法不属于预定义算法之一，则返回0。

6.1.12.4 异常抛出说明

无。

6.2 GBAUSignature

6.2.1 概述

GBAUSignature类是签名算法的基类。Signature算法的实现应扩展此类并实现所有抽象方法。

新增算法类型：

static final byte ALG_HMAC_SM3 HMAC

消息认证算法ALG_HMAC_SM3，此算法使用SM3作为哈希算法，按照RFC:2104中的步骤生成HMAC。

6.2.2 getInstance 方法 1

6.2.2.1 声明

创建 GBAUSignature类的实例。

public static GBAUSignature getInstance(byte algorithm,boolean externalAccess)

6.2.2.2 参数

algorithm：算法。

externalAccess：若为true，表示该实例可在多个卡应用实例间共享，且当 GBAUSignature 实例的所有者不是当前选中的卡应用时，也可通过Shareable接口访问该实例。若为false，实现不得为内部数据分配CLEAR_ON_ DESELECT类型的瞬态空间。

6.2.2.3 返回

对应算法的GBAUSignature对象。

6.2.2.4 异常抛出

若请求的算法不支持，则抛出CryptoException.NO_SUCH_ALGORITHM。

6.2.3 getInstance 方法 2

6.2.3.1 声明

使用所选的消息摘要算法、加密算法和填充算法创建GBAUSignature的一个实例。

public static GBAUSignature getInstance (byte messageDigestAlgorithm, byte cipherAlgorithm, byte paddingAlgorithm, boolean externalAccess)

在该类中仅使用在GBA_U过程中生成的内部 Ks_int_NAF 密钥（参见 3GPP TS 31.102 和 3GPP TS 33.220）。Ks_int_NAF 密钥为 256 位对称密钥，因此并非所有非对称密钥算法均支持。

6.2.3.2 参数

messageDigestAlgorithm：指定的消息摘要算法。

cipherAlgorithm：指定的加密算法，如SIG_CIPHER_AES_MAC128。

paddingAlgorithm：指定的填充算法。

externalAccess：若为true，表示该实例可在多个卡应用实例间共享，且当 GBAUSignature 实例的所有者不是当前选中的卡应用时，也可通过Shareable接口访问该实例。若为false，实现不得为内部数据分配CLEAR_ON_ DESELECT类型的瞬态空间。

6.2.3.3 返回

对应算法的GBAUSignature对象。

6.2.3.4 异常抛出

若请求的算法不支持，则抛出CryptoException.NO_SUCH_ALGORITHM。

6.2.4 init 方法 1

6.2.4.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥初始化GBAUSignature对象，该方法适用于不需要初始化参数或使用默认参数值的算法。

public abstract void init(byte theMode,byte[] adfAID, short adfAIDOff, short adfAIDLen,byte[] nafID,short nafIDOff,short nafIDLen)

6.2.4.2 参数

theMode：签名或验签模式。

adfAID：包含ADF的AID值的字节数组（例如USIM或ISIM的AID）。

adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。

adfAIDLen：ADF AID 值的字节长度。

nafID：包含NAF ID值的字节数组。

nafIDOff：nafID数组中NAF ID值的起始偏移量。

nafIDLen：NAF ID值的字节长度。

6.2.4.3 返回

无。

6.2.4.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；
——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API（访问条件详见第7章节）；
——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义（访问条件详见第7章节）；
——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，原因码CryptoException.ILLEGAL_USE，对应的情况如下：
——theMode参数为未定义值；
——密钥长度与getInstance()函数中定义的算法不兼容。
抛出NullPointerException：若adfAID或nafID为空。
抛出ArrayIndexOutOfBoundsException，对应情况如下：
——检查adfAIDOff或adfAIDLen时，访问超出adfAID数组边界；
——检查nafIDOff或nafIDLen时，访问超出nafID数组边界。

6.2.5 init 方法 2

6.2.5.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥和特定算法初始化GBAUCipher对象。此方法适用于不需要初始化参数或使用默认参数值的算法。

public abstract void init(byte theMode,byte[] adfAID,short adfAIDOff,short adfAIDLen,byte[] nafID,short nafIDOff,short nafIDLen,short keyLength)

6.2.5.2 参数

theMode：签名模式或验签模式。
adfAID：包含ADF AID值的字节数组（例如USIM或ISIM的AID）。
adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。
adfAIDLen：ADF AID 值的字节长度。
nafID：包含NAF ID值的字节数组。
nafIDOff：nafID数组中NAF ID值的起始偏移量。
nafIDLen：NAF ID值的字节长度。
keyLength：密钥长度 - 密钥的位数。有效的密钥位数长度取决于密钥类型。

6.2.5.3 返回

无。

6.2.5.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：
——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；
——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；
——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API（访问条件详见第7章节）；
——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义（访问条件详见第7章节）；
——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，原因码CryptoException.ILLEGAL_USE，对应的情况如下：

——theMode参数为未定义值；

——密钥长度与getInstance()函数中定义的算法不匹配。

抛出NullPointerException：若adfAID或nafID为空。

抛出ArrayIndexOutOfBoundsException，对应情况如下：

——检查adfAIDOff或adfAIDLen时，访问超出adfAID数组边界；

——检查nafIDOff或nafIDLen时，访问超出nafID数组边界。

6.2.6 init 方法 3

6.2.6.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥和特定算法初始化GBAUCipher对象。

public abstract void init(byte theMode,byte[] adfAID,short adfAIDOff,short adfAIDLen,byte[] nafID,short nafIDOff,short nafIDLen,byte[] bArray,short bOff,short bLen)

6.2.6.2 参数

theMode：签名模式或验签模式。

adfAID：包含ADF AID值的字节数组（例如USIM或ISIM的AID）。

adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。

adfAIDLen：ADF AID 值的字节长度。

nafID：包含NAF ID值的字节数组。

nafIDOff：nafID数组中NAF ID值的起始偏移量。

nafIDLen：NAF ID值的字节长度。

bArray：包含特定算法初始化信息的字节数组。

bOff：bArray数组中特定算法的数据的起始偏移量。

bLen：特定算法的参数数据的字节长度。

6.2.6.3 返回

无。

6.2.6.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；

——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API（访问条件详见第7章节）；

——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义（访问条件详见第7章节）；

——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，原因码CryptoException.ILLEGAL_USE，对应的情况如下：

——theMode参数为未定义值；

——密钥长度与getInstance()函数中定义的算法不匹配。

抛出NullPointerException：若adfAID或nafID为空。

抛出ArrayIndexOutOfBoundsException，对应情况如下：

——检查adfAIDOff或adfAIDLen时，访问超出adfAID数组边界；

——检查nafIDOff或nafIDLen时，访问超出nafID数组边界；

——检查bOff或bLen时，访问超出bArray数组边界。

6.2.7 init 方法 4

6.2.7.1 声明

使用NAF ID以及对应的Ks_int_NAF密钥和特定算法初始化GBAUCipher对象。

public abstract void init(byte theMode,byte[] adfAID,short adfAIDOff,short adfAIDLen,byte[] nafID,short nafIDOff,short nafIDLen,byte[] bArray,short bOff,short bLen，short keyLength)

6.2.7.2 参数

theMode：签名模式或验签模式。

adfAID：包含ADF AID值的字节数组（例如USIM或ISIM的AID）。

adfAIDOff：adfAID 数组中ADF AID值的起始偏移量。

adfAIDLen：ADF AID 值的字节长度。

nafID：包含NAF ID值的字节数组。

nafIDOff：nafID数组中NAF ID值的起始偏移量。

nafIDLen：NAF ID值的字节长度。

bArray：包含特定算法初始化信息的字节数组。

bOff：bArray数组中特定算法的数据的起始偏移量。

bLen：特定算法的参数数据的字节长度。

keyLength：密钥长度 - 密钥的位数。有效的密钥位数长度取决于密钥类型。

6.2.7.3 返回

无。

6.2.7.4 异常抛出

出现下述异常情况时，抛出GBAUException，具体原因码如下：

——GBA_U_BOOTSTRAP_NOT_DONE：Ks_int_NAF不可用（如GBA_U引导流程未执行）；

——GBA_U_NAF_DERIVATION_NOT_DONE：Ks_int_NAF不可用（如GBA_U NAF的派生流程未执行）；

——GBA_U_UNALLOWED_ACCESS：卡应用无权使用 API（访问条件详见第7章节）；

——GBA_U_INCORRECT_NAF_ID：若卡应用提供的NAF ID未在其访问条件中定义（访问条件详见第7章节）；

——GBA_U_INCORRECT_ADF_AID：若卡应用提供的ADF AID不支持GBA_U计算。

出现下述异常情况时，抛出javacard.security.CryptoException，原因码CryptoException.ILLEGAL_USE，对应的情况如下：

——theMode参数为未定义值；

——密钥长度与getInstance()函数中定义的算法不匹配。

NullPointerException：若adfAID或nafID为空。

抛出ArrayIndexOutOfBoundsException，对应情况如下：
——检查adfAIDOff或adfAIDLen时，访问超出adfAID数组边界；
——检查nafIDOff或nafIDLen时，访问超出nafID数组边界；
——检查bOff或bLen时，访问超出bArray数组边界。

6.2.8 update 方法

6.2.8.1 声明

使用init()方法所使用的NAF ID关联的 Ks_int_NAF 密钥来累积输入数据的签名。
public abstract void update(byte[] inBuff,short inOffset,short inLength)
该方法需要临时存储中间结果。如果输入数据长度不是块大小的整数倍，需要分配额外的内部存储空间来存储部分输入数据块。
仅当签名/验签所需的所有输入数据不能一次性提供在一个字节数组中时，才应使用此方法。
若所有输入数据位于单个字节数组中，建议使用 sign() 或 verify() 方法。
必须调用sign()或verify()来完成对通过一次或多次调用 update方法缓冲的剩余输入数据处理。
若inLength为0，此方法不执行任何操作。

6.2.8.2 参数

inBuff：待加密/解密的数据输入缓冲区。
inOffset：输入缓冲区中开始加密/解密的起始偏移量。
inLength：待加密/解密的字节长度。

6.2.8.3 返回

无。

6.2.8.4 异常抛出说明

javacard.security.CryptoException的原因码如下：
——CryptoException.UNINITIALIZED_KEY：密钥（Ks_int_NAF）未初始化；
——CryptoException.INVALID_INIT：GBAUCipher 对象未正确初始化；
——CryptoException.ILLEGAL_USE：输入消息长度不被支持，或消息值大于等于模数；
——NullPointerException：若inBuff为空；
——ArrayIndexOutOfBoundsException：若对inBuff的检查导致数组越界访问。

6.2.9 sign 方法

6.2.9.1 声明

使用init方法指定的NAF ID及关联的Ks_int_NAF密钥，对所有/最后输入数据执行签名操作。
public abstract short sign(byte[] inBuff,short inOffset,short inLength,byte[] sigBuff,short sigOffset)
调用此方法后，GBAUSignature对象会重置为其先前通过调用 init()方法初始化时的状态，可供其他消息进行签名使用。在CBC模式下使用AES时所使用的初始向量（IV）将被重置为0。
输入和输出缓冲区的数据可能会重叠。
除返回结果，此方法还会将结果设置到内部状态中，如果平台支持的话，可以通过javacardx.security.SensitiveResult 类的断言方法来再次检查该状态。

6.2.9.2 参数

inBuff：待加密/解密的数据输入缓冲区。
inOffset：输入缓冲区中开始加密/解密的起始偏移量。
inLength：待加密/解密的字节长度。
sigBuff：用于存储签名数据的输出缓冲区。
sigOffset：在sigBuff中开始存放签名数据的位置偏移量。

6.2.9.3 返回

签名输出在sigBuff中所占的字节数。

6.2.9.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：
——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；
——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化；
——CryptoException.ILLEGAL_USE：如果消息值不被 GBAUSignature 算法所支持，或者消息值一致性检查失败；
——NullPointerException：若inBuff或sigBuff 为空；
——ArrayIndexOutOfBoundsException：inOffset或inLength的检查操作导致访问超出inBuff 数组边界；
——ArrayIndexOutOfBoundsException：sigOffset检查操作导致超出sigBuff数组边界。

6.2.10 verify 方法

6.2.10.1 声明

使用init方法指定的NAF ID及关联的Ks_int_NAF密钥，对所有/最后输入数据的签名进行验证。

public abstract boolean verify(byte[] inBuff,short inOffset,short inLength,byte[] sigBuff,short sigOffset,short sigLength)

调用此方法后，GBAUSignature对象会重置为其先前通过调用 init()方法初始化时的状态，可供其他消息进行签名数据验证使用。在 CBC 模式下使用AES时所使用的初始向量（IV）将被重置为0。

除了返回一个布尔值结果外，该方法还会将结果设置到内部状态中，如果平台支持的话，可以通过SensitiveResult类的断言方法来再次检查该状态。

6.2.10.2 参数

inBuff：待加密/解密的数据输入缓冲区。
inOffset：输入缓冲区中开始加密/解密的起始偏移量。
inLength：待加密/解密的长度。
sigBuff：用于存储签名数据的输出缓冲区。
sigOffset：在sigBuff中开始存放签名数据的位置偏移量。
sigLength：签名数据长度。

6.2.10.3 返回

如果签名验证通过，则返回“1”，否则返回“0”。如果签名长度与该签名算法不一致，则返回“0”。

6.2.10.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化；

——返回CryptoException.ILLEGAL_USE，对应情况如下：

- 如果GBAUSignature算法未对消息进行填充，并且消息未按块对齐；
- 如果GBAUSignature算法未对消息进行填充，且在inBuff中未提供输入数据，也未通过update()方法提供输入数据；
- 如果消息值不被GBAUSignature算法所支持，或者消息值一致性检查失败；
- 如果此GBAUSignature算法包含消息恢复功能；

——NullPointerException：若inBuff或sigBuff 为空；

——ArrayIndexOutOfBoundsException：inOffset或inLength的检查操作导致访问超出inBuff数组边界；

——ArrayIndexOutOfBoundsException：sigOffset检查操作导致超出sigBuff数组边界。

6.2.11 setInitialDigest 方法

6.2.11.1 声明

该方法会直接在初始化起始哈希值，而非使用GBAUSignature类所采用的默认值。起始哈希值代表的是通过相同算法计算出的、消息前半部分的先前哈希值。消息的其余字节必须通过update()、sign()或verify()方法传递给此GBAUSignature对象，以生成或验证签名。

public abstract void setInitialDigest(byte[] initialDigestBuf,short initialDigestOffset,short initialDigestLength,byte[] digestedMsgLenBuf,short digestedMsgLenOffset,short digestedMsgLenLength)

消息第一部分的字节长度的最大允许值取决于具体算法。

该方法在底层签名算法在应用加密原语之前未计算出独特的消息摘要值时会抛出异常。AES、HMAC和KOREAN SEED算法会抛出异常。

6.2.11.2 参数

initialDigestBuf：包含起始哈希值的输入缓冲区，该值代表了（使用相同的算法）消息前半部分之前计算出的哈希值。

initialDigestOffset：将数据偏移至初始摘要缓冲区数组中，该数组的起始位置即为摘要值数据所在的位置。

initialDigestLength：初始摘要缓冲区数组中数据的长度。

digestedMsgLenBuf：包含消息第一部分中字节数的字节数组，该部分已先进行哈希处理以获取指定的起始摘要值。

digestedMsgLenOffset：在已处理消息长度缓冲区中，用于标识已处理长度的起始位置（从该位置开始的字节（共 digestedMsgLenLength 个字节）被连接起来以形成实际的已处理消息长度值）。

digestedMsgLenLength：被处理后的长度的字节长度。

6.2.11.3 返回

无。

6.2.11.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化；

——返回CryptoException.ILLEGAL_VALUE，对应情况如下

- 如果参数“initialDigestLength”与该算法的中间哈希值大小不相等；
- 如果消息中先前已进行哈希处理的部分中的字节数为0或者不是算法块大小的倍数，或者大于该算法所支持的最大长度；

——返回CryptoException.ILLEGAL_USE，对应情况如下：

- 如果签名算法在应用加密原语之前未计算出独特的消息摘要值；
- 如果该签名算法包含消息恢复功能。

6.2.12 signPreComputedHash 方法

6.2.12.1 声明

生成预计算哈希数据的签名。

public abstract short signPreComputedHash(byte[] hashBuff, short hashOffset, short hashLength, byte[] sigBuff, short sigOffset)

调用此方法还会将GBAUSignature对象重置为其先前通过调用init()方法初始化时的状态，并可再次用于对另一个预先计算的哈希值进行签名。

该方法在底层签名算法在应用加密原语之前未计算出独特的消息摘要值时会抛出异常。AES、HMAC和KOREAN SEED算法会抛出异常。

之前通过调用更新方法累积的任何数据都会被丢弃。

哈希和输出缓冲区的数据可能会重叠。

除返回结果外，该方法还会将结果设置到内部状态中，如果平台支持的话，可以使用javacardx.security.SensitiveResult 类的断言方法来再次检查该状态。

6.2.12.2 参数

hashBuff：用于签名的预计算哈希值的输入缓冲区。

hashOffset：哈希值在缓冲区中的起始位置偏移量。

hashLength：哈希值的字节长度。

sigBuff：用于存储签名数据的输出缓冲区。

sigOffset：在 sigBuff 中开始存放签名数据的位置偏移量。

6.2.12.3 返回

签名输出在sigBuff中所占的字节数。

6.2.12.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化；

——返回CryptoException.ILLEGAL_USE，对应情况如下：

- 如果哈希长度值与算法的消息摘要长度不相等；

- 如果GBAUSignature算法包含消息恢复功能；
- 如果GBAUSignature算法在应用加密原语之前未计算出独特的消息摘要值。

6.2.13 verifyPreComputedHash 方法

6.2.13.1 声明

验证预计算哈希数据的签名。

public abstract boolean verifyPreComputedHash(byte[] hashBuff, short hashOffset, short hashLength, byte[] sigBuff, short sigOffset, short sigLength)

调用此方法还会将GBAUSignature对象重置为其先前通过调用init()方法初始化时的状态，并可再次用于对另一个预先计算的哈希值进行验证。

AES和KOREAN SEED算法的CBC模式中使用的初始向量（IV）将被重置为 0。

如果底层的签名算法在应用加密原语之前未能计算出独特的消息摘要值,该方法就会抛出异常。AES和KOREAN SEED算法会抛出异常。

之前通过调用更新方法累积的任何数据都会被丢弃。

哈希和输出缓冲区的数据可能会重叠。除返回布尔值结果外,该方法还会将结果设置到内部状态中，如果平台支持的话，可以通过SensitiveResult类的断言方法来再次检查该状态。

6.2.13.2 参数

hashBuff：用于签名的预计算哈希值的输入缓冲区。

hashOffset：哈希值在缓冲区中的起始位置偏移量。

hashLength：哈希值的字节长度。

sigBuff：用于存储签名数据的输出缓冲区。

sigOffset：在 sigBuff 中开始存放签名数据的位置偏移量。

sigLength：签名数据的字节长度。

6.2.13.3 返回

如果签名验证通过，则返回“1”，否则返回“0”。如果“sigLength”与GBAUSignature算法不一致，则返回“0”。

6.2.13.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化；

——返回CryptoException.ILLEGAL_USE，对应情况如下：

- 如果哈希长度值与算法的消息摘要长度不相等；
- 如果GBAUSignature算法包含消息恢复功能；
- 如果GBAUSignature算法在应用加密原语之前未计算出独特的消息摘要值。

6.2.14 getAlgorithm 方法

6.2.14.1 声明

获取签名算法。从 javacard.security.Signature 类的 ALG_* 常量中列出的预定义代码，例如 ALG_ AES_MAC4_NOPAD 。

public abstract byte getAlgorithm()

6.2.14.2 参数

无。

6.2.14.3 返回

算法代码在javacard.security.Signature类中进行了定义;如果所选算法并非预定义的算法之一，则返回值为 0 。

6.2.14.4 异常抛出说明

无。

6.2.15 getCipherAlgorithm 方法

6.2.15.1 声明

获取加密算法。从 javacard.security.Signature 类的 SIG_CIPHER_* 常量中列出的预定义代码，例如SIG_CIPHER_AES_MAC128。

public abstract byte getCipherAlgorithm()

6.2.15.2 参数

无。

6.2.15.3 返回

算法代码在javacard.security.Signature类中进行了定义;如果所选算法并非预定义的算法之一，则返回值为0。

6.2.15.4 异常抛出说明

无。

6.2.16 getPaddingAlgorithm 方法

6.2.16.1 声明

获取填充算法。从javacardx.crypto.Cipher类的 PAD_* 常量中列出的预定义代码,例如 PAD_NULL。

public abstract byte getPaddingAlgorithm()

6.2.16.2 参数

无。

6.2.16.3 返回

算法代码在javacard.security.Signature类中进行了定义;如果所选算法并非预定义的算法之一，则返回值为0。

6.2.16.4 异常抛出说明

无。

6.2.17 getMessageDigestAlgorithm 方法

6.2.17.1 声明

获取消息摘要算法。从javacard.security.MessageDigest类的 ALG_* 常量中列出的预定义代码，例如 ALG_NULL。

public abstract byte getMessageDigestAlgorithm()

6.2.17.2 参数

无。

6.2.17.3 返回

消息摘要算法代码在 javacard.security.MessageDigest 类中定义；如果所使用的算法并非预定义的算法之一，则返回值为0。

6.2.17.4 异常抛出说明

无。

6.2.18 getLength 方法

6.2.18.1 声明

获取签名数据长度。

public abstract short getLength()

6.2.18.2 参数

无。

6.2.18.3 返回

返回签名数据长度。

6.2.18.4 异常抛出说明

抛出javacard.security.CryptoException，原因码如下：

——CryptoException.UNINITIALIZED_KEY：若Ks_int_NAF密钥未初始化；

——CryptoException.INVALID_INIT：若GBAUSignature对象未初始化。

6.3 GBAUException

6.3.1 概述

GBAUException类定义GBAUCipher和GBAUSignature类在出现异常情况时抛出特定异常。

6.3.2 字段摘要

GBAUException字段摘要见表2。

表2　字段摘要说明

修饰符与类型	字段	描述
static final short	GBA_U_BOOTSTRAP_NOT_DONE	该原因码（= 1）用于指示因GBA引导流程（GBA Bootstrap）未完成，导致加密/签名操作无法执行。
static final short	GBA_U_INCORRECT_ADF_AID	该原因码（= 5）用于指示因卡应用请求使用的ADF AID不支持 GBA_U，导致加密/签名操作无法执行。
static final short	GBA_U_INCORRECT_NAF_ID	该原因码（= 4）用于指示因卡应用请求使用的NAF_ID不在允许范围内,导致加密/签名操作无法执行。
static final short	GBA_U_NAF_DERIVATION_NOT_DONE	该原因码（= 2）用于指示因GBA NAF派生流程未完成，导致加密/签名操作无法执行。
static final short	GBA_U_UNALLOWED_ACCESS	该原因码(=3)用于指示应用不允许使用USIM API，导致加密/签名操作无法执行。）
static void	throwIt(short reason)	抛出带有指定原因的 GBAUException 类的 JCRE 实例

6.3.3　字段详情

GBAUExecption字段详情见表3。

表3　字段详情说明

字段	详情
GBA_U_BOOTSTRAP_NOT_DONE	public static final short GBA_U_BOOTSTRAP_NOT_DONE该原因码（= 1）用于指示因GBA引导流程（GBA Bootstrap）未完成，导致加密/签名操作无法执行。
GBA_U_NAF_DERIVATION_NOT_DONE	public static final short GBA_U_NAF_DERIVATION_NOT_DONE该原因码（= 2）用于指示因GBA NAF 派生流程未完成，导致加密/签名操作无法执行。
GBA_U_UNALLOWED_ACCESS	public static final short GBA_U_UNALLOWED_ACCESS该原因码（= 3）用于指示应用不被允许使用USIM中指定的 API，导致加密/签名操作无法执行。
GBA_U_INCORRECT_NAF_ID	public static final short GBA_U_INCORRECT_NAF_ID该原因码（= 4）用于指示应用请求使用的 NAF_ID不在允许范围内，导致加密/签名操作无法执行。
GBA_U_INCORRECT_ADF_AID	public static final short GBA_U_INCORRECT_ADF_AID该原因码（= 5）用于指示应用请求使用的ADF AID不支持GBA_U，导致加密/签名操作无法执行。

6.3.4　构造方法详情

GBAUExecption构造方法详情见表4。

表4　构造方法详情

构造方法	详情
GBAUException	public GBAUException(short reason)使用指定的原因创建 GBAUException 实例。为节省资源，建议使用throwIt()方法复用此类的 JCRE 实例。参数：reason - 异常原因

6.3.5 方法详情

GBAUExecption中throwIt方法详情见表5。

表5 方法详情

方法	详情
throwIt	public static void throwIt(short reason) throws GBAUException 抛出带有指定原因的GBAUException 类的 JCRE 实例。参数：reason - 异常原因 抛出：GBAUException - 始终抛出此异常。

6.4 算法类型

GBA_U API中方法会用到的算法参数和支持的算法类型所对应的名称以及引用的库名详见表6。

表6 GBA_U API方法的算法参数和支持的算法类型说明

参数	算法名称	引用库（javacard 3.1.0）
GBAUCipher		
algorithm	ALG_SM4_CBC_NOPAD	javacardx.crypto.Cipher
	ALG_SM4_CBC_ISO9797_M1	
	ALG_SM4_CBC_ISO9797_M2	
	ALG_SM4_CBC_PKCS5	
	ALG_AES_BLOCK_128_CBC_NOPAD	
	ALG_KOREAN_SEED_CBC_NOPAD	
	ALG_AES_CBC_ISO9797_M1	
	ALG_AES_CBC_ISO9797_M2	
	ALG_AES_CBC_PKCS5	
	ALG_AES_BLOCK_128_ECB_NOPAD	
	ALG_KOREAN_SEED_ECB_NOPAD	
	ALG_AES_ECB_ISO9797_M1	
	ALG_AES_ECB_ISO9797_M2	
	ALG_AES_ECB_PKCS5	
cipherAlgorithm	CIPHER_AES_CBC	
	CIPHER_KOREAN_SEED_CBC	
	CIPHER_AES_ECB	
	CIPHER_KOREAN_SEED_ECB	
	CIPHER_SM4_ECB	
	CIPHER_SM4_CBC	
paddingAlgorithm	PAD_NULL	
	PAD_NOPAD	
	PAD_ISO9797_M1	
	PAD_ISO9797_M2	
	PAD_PKCS5	
GBAUSignature		

表6 GBA_U API方法的算法参数和支持的算法类型说明（续）

参数	算法名称	引用库（javacard 3.1.0）
algorithm	ALG_HMAC_SM3	javacard.security.Signature
	ALG_KOREAN_SEED_MAC_NOPAD	
	ALG_AES_MAC_128_NOPAD	
	ALG_HMAC_SHA_256	
	ALG_HMAC_SHA_384	
	ALG_HMAC_SHA_512	
cipherAlgorithm	SIG_CIPHER_AES_MAC128	
	SIG_CIPHER_HMAC	
	SIG_CIPHER_KOREAN_SEED_MAC	
messageDigestAlgorithm	ALG_NULL	javacard.security.MessageDigest
	ALG_SHA_256	
	ALG_SHA_384	
	ALG_SHA_512	
	ALG_SM3	
paddingAlgorithm	PAD_NULL	javacardx.crypto.Cipher
	PAD_NOPAD	

7 GBA_U API 访问控制机制

在对加密对象进行初始化时（即调用GBAUCipher、GBAUSignature类中的init()方法），需要先完成访问权限的检查：

——参数adfAID、adfAIDOff和adfAIDLen对应USIM或ISIM（参照TS 31.130中相关定义）；

——在相应的ADF中存在EFAC_GBAUAPI，详见表7；

——至少有一条EFAC_GBAUAPI记录与以下内容匹配：

- 调用方applet的AID值（即adfAID参数）；
- NAF_ID值的nafID、nafIDOff和nafIDLen参数。

如果所有条件都满足，则会初始化并返回加密对象，否则将抛出一个GBAUException异常，原因代码为GBA_U_UNALLOWED_ACCESS。

表7　访问控制文件格式

<table>
<tr><td colspan="2">文件标识符：’6Fxx’（注 1）</td><td colspan="2">线性记录文件</td><td>可选</td></tr>
<tr><td colspan="3">记录长度：Z＞7 bytes</td><td colspan="2">更新频率：低</td></tr>
<tr><td colspan="5">访问条件：
READ　　ADM
UPDATE　　ADM
DEACTIVATE　　ADM
ACTIVATE　　ADM</td></tr>
<tr><td>字节</td><td colspan="2">描述</td><td>M/O</td><td>长度</td></tr>
<tr><td>1 to Z</td><td colspan="2">Applet NAF 访问控制 TLV 对象</td><td>M</td><td>Z bytes</td></tr>
<tr><td colspan="5">注：若为USIM卡，则文件标识符为6Fxx，若为ISIM，则文件标识符为6F0A。</td></tr>
</table>

Applet NAF 访问控制 TLV 对象值描述见表 8。

表 8　Applet NAF 访问控制 TLV 对象值描述

描述	标签值
Applet NAF 访问控制 TLV 对象	’80’

Applet NAF 访问控制 TLV 对象详细描述见表 9。

表 9　Applet NAF 访问控制 TLV 对象详细描述

描述	值	M/O	长度(字节)
Applet NAF 访问控制 TLV 对象标签	’80’	M	1
长度	注 1	M	注 2
AID 长度	X	M	注 2
AID 值	--	M	5-16 字节
NAF_ID 长度	Y	M	注 2
NAF_ID 值	--	M	Y
注 1:TLV 对象的总长度。 注 2:长度编码应遵循 ISO/IEC 8825-1。			

附 录 A
（资料性）
GBA_U 派生密钥Ks_int_NAF生成过程

GBA_U 派生密钥Ks_int_NAF是通过GBA_U认证生成，具体可参见3GPP TS 33.220、3GPP TS 31.102或3GPP TS 31.103。

GBA_U认证机制包括引导模式和NAF推导模式两个流程，执行引导模式流程后再执行NAF推导模式流程，具体要求如下：

a） 引导模式（Bootstrapping Mode）：终端中的SIM与网络侧的引导服务功能（BSF）通过AKA协议进行双向认证，生成并共享主密钥Ks；

b） NAF推导模式（NAF Derivation Mode）：基于引导模式生成的主密钥Ks，SIM基于Ks和NAF标识生Ks_int_NAF密钥，存储于SIM内部。

附 录 B
（规范性）
GBA_U APIpackage信息

GBA_U API的package名为uicc.usim.gba_u，AID值为A000000087 1005 FFFF FFFF 89 13 500000。

参 考 文 献

[1] 3GPP TS 31.130 (U)SIM Application Programming Interface (API);(U)SIM API for Java™ Card

电信终端产业协会团体标准

基于 SIM 卡的 GBA_U 派生密钥 API 技术要求

T/TAF 333—2026

*

电信终端产业协会发布
地址：北京市西城区新街口外大街 28 号
电话：010-82052809
电子版下载网址：www.taf.org.cn