ICS 35.040
L 80
备案号：

中华人民共和国密码行业标准

GM/T 0058-2018

可信计算 TCM服务模块接口规范

Trusted computing TCM service module interface specification

（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

可信计算标准体系包含以下标准：

GM/T 0011-2012 可信计算 可信密码支撑平台功能与接口规范

GM/T 0012-2012 可信计算 可信密码模块接口规范

GM/T 0013-2012 可信计算 可信密码模块接口符合性测试规范

GM/T XXXX-XXXX 可信计算 TCM服务模块接口规范

上述四个标准中，本标准的接口介于GM/T 0012-2012之上，向上为应用程序提供接口调用，中间定义了如何实现服务模块接口，向下调用GM/T 0012-2012中的接口，本标准以GM/T 0011-2012为基础及核心，技术内容基本未做改动，GM/T 0013-2012为可信密码模块产品的符合性检测提供依据，四个规范形成一套完整的可信计算体系标准。

本标准根据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：国民技术股份有限公司、联想控股有限公司、同方股份有限公司、中国科学院软件所、北京兆日技术有限责任公司、瑞达信息安全产业股份有限公司、长春吉大正元信息技术股份有限公司、方正科技集团股份有限公司、北京信息科技大学、中国长城计算机深圳股份有限公司、成都卫士通信息产业股份有限公司、无锡江南信息安全工程技术中心、中国人民解放军国防科学技术大学、北京工业大学。

本标准主要起草人：吴秋新、杨贤伟、范琴、邹浩、余发江、宁晓魁、王梓、郑必可、刘鑫、林洋、李伟平、尹洪兵、徐震、严飞、付月朋、明明、刘韧、李丰、许勇、贾兵、王蕾、顾健、何长龙、秦宇。

本标准凡涉及密码算法的相关内容，按国家有关法规实施；凡涉及到采用密码技术解决保密性、完整性、真实性、不可否认性需求的应遵循密码相关国家标准和行业标准。

可信计算 TCM 服务模块接口规范

1 范围

本标准规定了TCM服务模块的组成和接口标准，包含TSP、TCS和TDDL,是面向TCM应用层的接口标准。
本标准适用于基于TCM的应用开发。

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 SM3密码杂凑算法
GB/T 32907-2016 SM4分组密码算法
GB/T 32918.2-2016 SM2椭圆曲线公钥密码算法 第2部分:数字签名算法
GB/T 32918.4-2016 SM2椭圆曲线公钥密码算法 第4部分:公钥加密算法
GM/T 0005-2012 随机性检测规范
GM/T 0009-2012 SM2密码算法使用规范
GM/T 0015-2012 基于SM2密码算法的数字证书格式规范

3 术语和定义

以下术语和定义适用于本文件。

3.1

部件 component
计算系统中可被度量的硬件和/或软件模块。

3.2

存储主密钥 storage master key
用于保护平台身份密钥和用户密钥的主密钥，是可信存储根的一种实现形式。

3.3

对象 object
可信计算密码支撑平台内可以被实体访问的各类资源，包括密钥数据、运行环境数据、敏感数据等。

3.4

可信计算平台 trusted computing platform
构建在计算系统中，用于实现可信计算功能的支撑系统。

3.5

可信计算密码支撑平台 cryptographic support platform for trusted computing

可信计算平台的重要组成部分，包括密码算法、密钥管理、证书管理、密码协议、密码服务等内容，为可信计算平台自身的完整性、身份可信性和数据安全性提供密码支持。其产品形态主要表现为可信密码模块和可信密码服务模块。

3.6

可信报告 trust report
一种用于标识平台身份、平台配置和平台证明的报告。

3.7

可信度量根 root of rrust for measurement
一个可信的完整性度量单元，是可信计算平台内进行可信度量的基础。

3.8

可信存储根 root of trust for storage
是一种通用安全机制，是可信计算平台内进行可信存储的基础。

3.9

可信报告根 root of trust for reporting
指密码模块密钥，是可信计算平台内进行可信报告的基础。

3.10

可信密码模块 trusted cryptography module
是可信计算平台的硬件模块，为可信计算平台提供密码运算功能，具有受保护的存储空间。

3.11

TCM 服务模块 tcm service module
可信计算密码支撑平台内部的软件模块，为对平台外部提供访问可信密码模块的软件接口。

3.12

可信方 trusted party
提供可信证明的机构，包含可信第三方和主管方。

3.13

密码模块密钥 tcm dndorsement key
可信密码模块的背书密钥。

3.14

密钥管理中心 key management centre
用于密钥生成和管理的软硬件系统。

3.15

平台配置寄存器 platform configuration register
可信密码模块内部用于存储平台配置值的存储单元。

3.16

平台身份密钥 platform identity key
可信密码模块的身份密钥。

3.17

平台加密密钥 platform encryption key
可信密码模块中与平台身份密钥关联的加密密钥。

3.18

双证书 dual certificate
包括签名证书和加密证书，分别用于签名和数据加密，由可信方一同签发。

3.19

实体 entity
访问密码支撑平台资源的应用程序和用户。

3.20

完整性度量 integrity measurement
使用密码杂凑算法对被度量对象计算其杂凑值的过程。

3.21

完整性度量值 integrity measurement value
部件被度量后得到的杂凑值。

3.22

完整性基准值 predefined integrity value
部件在可信状态下被度量得到的杂凑值。该值可作为完整性校验的基准。

3.23

信任链 trusted chain
在计算系统启动和运行过程中，使用完整性度量方法在部件之间所建立的信任传递关系。

4 缩略语

下列缩略语适用于本文件。

EK 可信计算密码模块密钥（TCM Endorsement Key）
HMAC 带密钥的消息验证码算法（the keyed-hash message authentication code）
NV 非易失性（Non-Volatility）
PCR 平台配置寄存器（Platform Configuration Register）
PEK 平台加密密钥（Platform Encryption Key）
PIK 平台身份密钥（Platform Identity Key）
SMK 存储主密钥（Storage Master Key）
TCM 可信密码模块（Trusted Cryptography Module）

TSM　TCM服务模块（TCM service module）
TSP　TCM应用服务（TCM Service Provider）
TCS　TCM核心服务（TCM Core Services）
TDD　TCM设备驱动（TCM Device Drive r ）
TDDL　TCM设备驱动库（TCM Device Driver Library）
TDDLI　TCM设备驱动库接口（TCM Device Driver Library Interface）

5　软件架构

TCM服务模块软件架构如图1所示，包括TCM应用服务、TCM核心服务和TCM设备驱动库三个组成部分。

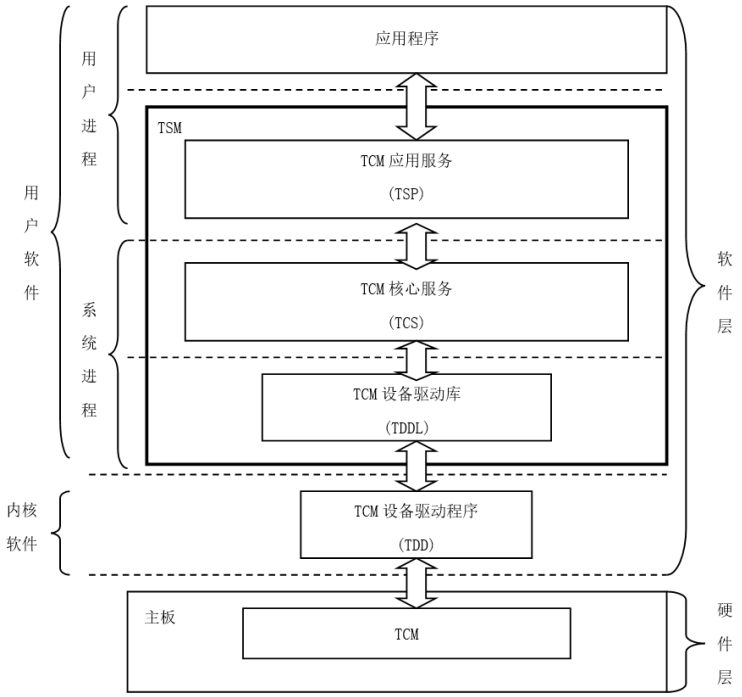

图 1 TCM 服务模块软件架构图

a)　TCM 应用服务(TCM Service Provider：TSP)

TSP 向应用程序提供 TCM 的服务，使应用程序只关注它本身的特性，通过提供上层的 TCM 函数接口，使 TSP 执行 TCM 提供的可信函数。TSP 还提供了一些方便功能操作的辅助函数，这些函数不是由 TCM 提供，如：签名验证功能。

TSP 位于应用程序进程内，使每个应用程序看起来像拥有一个自己的 TSP。

多进程的操作系统中，会有多个 TSP 实例运行在平台上。

一个 TSM 的执行需要有 TSP：

1)它们负责对应用程序间信息和数据的传输提供保护；

2)提供 C 语言接口或者能被各个平台调用的通用接口，和对应用程序的动态链接或静态连接；

3)运行在 Windows 操作系统的 TSM 也可以提供 COM 接口。

b)　TCM 核心服务(TCM Core Services：TCS)

TCS 位于 TSM 服务提供者(TSP)层和 TCM 设备驱动库（TDDL）层之间，以系统服务的形式存在，为 TSP 等上层应用提供 TCM 使用和密钥管理等功能接口。

TCS 按功能的不同可分为：

基本信息管理、密钥管理、密钥缓存管理、事件管理、授权操作、完整性操作、迁移操作、密码操作、身份证书操作、设备操作、密钥协商类 11 个模块，其中基本信息管理、密钥管理和事件管理同属于 TCS 管理器，密钥缓存管理、授权操作、完整性操作、迁移操作、密码操作、身份证书操作、设备操作同属于 TCM 操作。

c) TCM 设备驱动库(TDDL)

TDDL 位于 TCM 核心服务（TCS）层和 TCM 设备驱动（TDD）层之间，主要目的是在 TDD 之上提供一个标准的接口，屏蔽各设备 I/O 控制信息的差异，完成信息在用户软件和内核软件的传递。

本标准以 C 语言为例编制相关函数、接口进行说明。

6 TCM 应用服务

6.1 类定义

TCM 应用服务定义如下类：

a)上下文管理类(Context class)

上下文管理类包含关于 TSM 对象的执行环境的信息

b)策略管理类(Policy class)

策略管理类用于为不同用户应用程序配置相应的安全策略与行为。

应用程序通过策略管理为授权机制提供专门的授权秘密信息的处理操作（如回调，生命周期）。

c)TCM 管理类(TCM class)

TCM 管理类用于表示 TCM 的所有者。TCM 的所有者可以看成是 PC 环境中的系统管理员。因此，每个上下文仅有一个 TCM 管理类的实例。这个对象自动与一个策略对象相关联，用于处理 TCM 所有者的授权数据。此外，它还提供一些基本控制和报告的功能。

d)密钥管理类(Key class)

密钥管理类用于表示 TSM 密钥管理功能的入口。每个密钥对象的实例代表 TSM 密钥树中一个具体的密钥节点。每个需授权的密钥对象，需要分配一个策略对象，用于管理授权秘密信息。

e)数据加解密类(Encrypted Data class)

该类用于将外部（例如用户，应用程序）产生的数据与系统关联起来（与平台或 PCR 绑定），或者用于为外部提供数据加密/解密服务。

进行授权处理时，可以给该类分配一个策略对象。

f)PCR 操作类(PCR Composite class)

PCR 操作类可用于建立系统平台的信任级别。该类提供了对 PCR 进行选择、读、写等操作的简便方法。

所有需要 PCR 信息的函数，在其参数表中都使用了这个类的对象句柄。

g)NV 存储管理类(NV Store class)

NV 存储管理类用于管理 TCM 的非易失性存储区域中存储属性信息。用于 NV 区域的定义、读、写和释放。

该类建立 NV 存储区的大小、索引、各种读写授权，其中授权可以基于 PCR 值或者授权数据，但不能基于 Locality。

h)杂凑计算类(Hash class)

杂凑计算类为数字签名操作提供了一种密码学安全的方法。

i)密钥协商类(Exchange Key class)

密钥协商类用于执行密钥交换协议操作。

6.2 类与对象的关系

工作对象可再分为授权工作对象和非授权工作对象。非授权工作对象包括 PCR 组合对象和杂凑对象，见图 2。授权对象包括 TCM 对象、密钥对象、NV 存储对象、迁移数据对象和加密数据对象，见图 3。

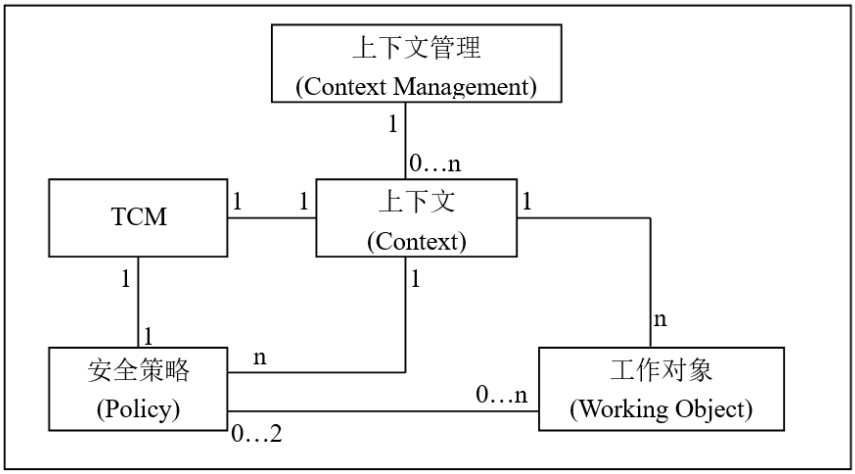

图 2 安全策略与工作对象之间的关系

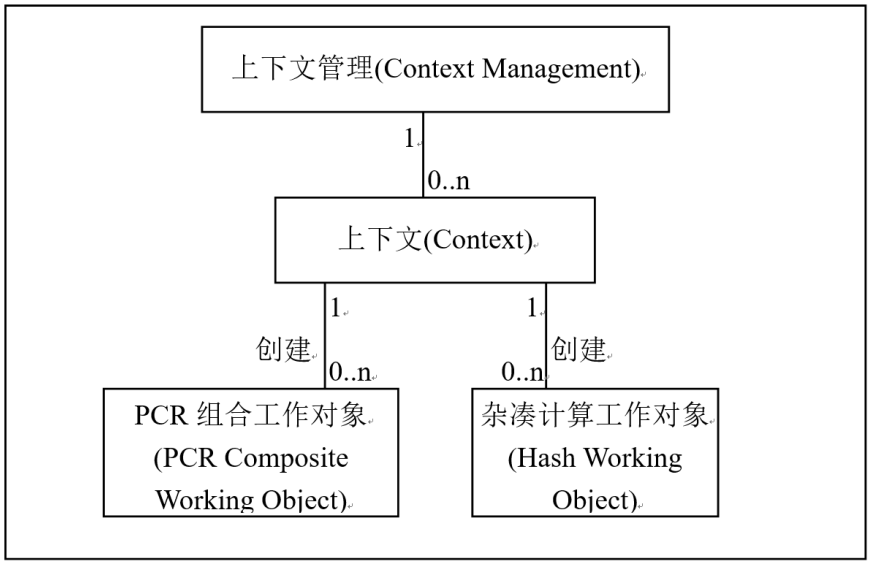

图 3 非授权工作对象

在非授权工作对象中，可以创建多个 PCR 组合对象和杂凑对象。

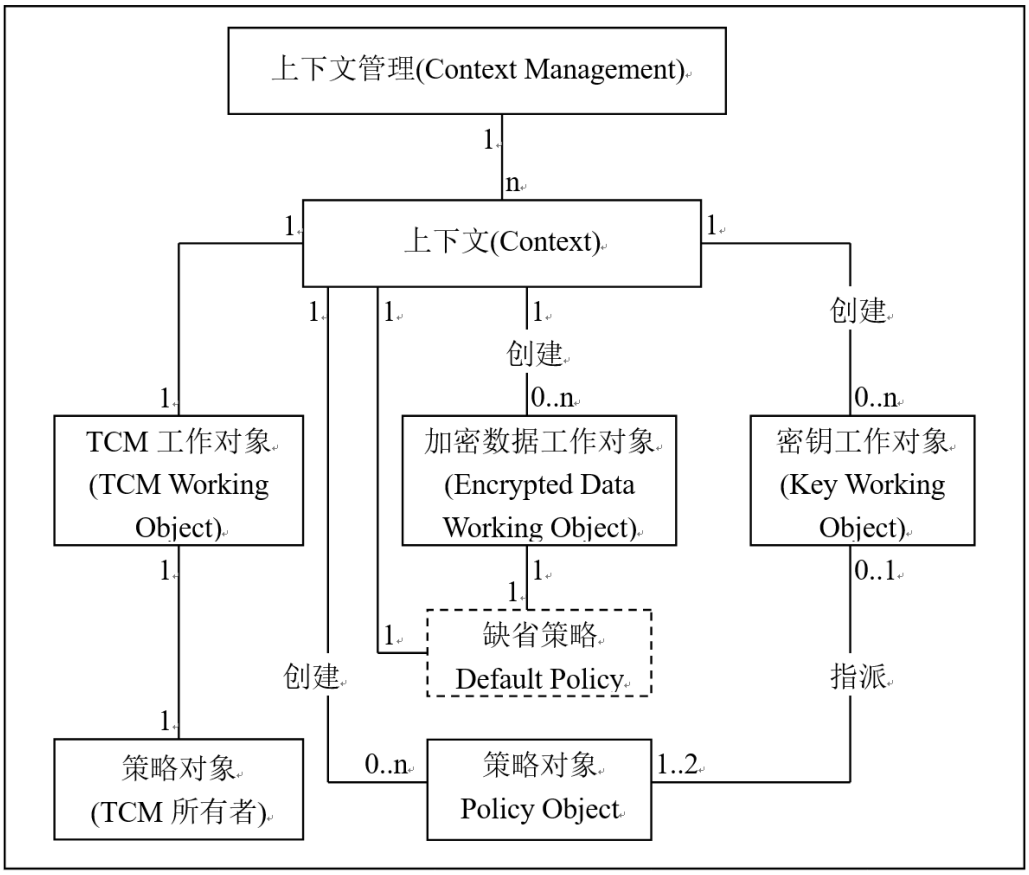

图 4 需授权的工作对象之间关系

调用程序（用户）可能只为使用的每个策略提供一次授权数据。一个策略可能分配给多个对象，比如密钥对象，加密数据对象，或者 TCM 对象。这些对象中的每一个都将使用分配给它的策略对象的内部函数来处理授权 TCM 命令,关系见图 4。

6.3 接口

6.3.1 通用接口

6.3.1.1 Tspi_SetAttribUint32

功能描述：

本函数设置对象的32-bit 属性。如果请求的数据长度小于UINT32，必须将数据转换成32-bit长度的数据。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
    TSM_HOBJECT     hObject,        // in
    TSM_FLAG        attribFlag,     // in
    TSM_FLAG        subFlag,        // in
    UINT32          ulAttrib        // in
```

```
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　需要设置的属性标记。

——subFlag　　需要设置的子属性标记。

——ulAttrib　　属性设置的值。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.1.2 Tspi_GetAttribUint32

功能描述：

本函数获取对象的32-bit 属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
    TSM_HOBJECT     hObject,        // in
    TSM_FLAG        attribFlag,     // in
    TSM_FLAG        subFlag,        // in
    UINT32*         pulAttrib       // out
);
```

输入参数描述：

——hObject　　需要查询属性的对象句柄。

——attribFlag　需要查询的属性标记。

——subFlag　　需要查询的子属性标记。

输出参数描述：

——pulAttrib　指向查询到的属性值。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.1.3 Tspi_SetAttribData

功能描述：

本函数设置对象的非32-bit 属性。 属性数据的结构和大小依属性而定。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
    TSM_HOBJECT     hObject,            // in
    TSM_FLAG        attribFlag,         // in
    TSM_FLAG        subFlag,            // in
    UINT32          ulAttribDataSize,   // in
    BYTE*           rgbAttribData       // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　　需要设置的属性标记。

——subFlag　　需要设置的子属性标记。

——ulAttribDataSize　　prgbAttribData参数的大小（以字节为单位）。

——rgbAttribData　　此参数指向一个存放设置属性值的缓冲区。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.1.4 Tspi_GetAttribData

功能描述：

本函数获取对象的非32-bit 属性。 属性数据的结构和大小依属性而定。为属性数据所分配的内存块必须用Tspi_Context_FreeMemory来释放。

接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
    TSM_HOBJECT         hObject,            // in
    TSM_FLAG            attribFlag,         // in
    TSM_FLAG            subFlag,            // in
    UINT32*             pulAttribDataSize,  // out
    BYTE**              prgbAttribData      // out
);
```

输入参数描述：

——hObject　　需要获取属性的对象句柄。
——attribFlag　需要获取的属性标记。
——subFlag　　需要获取的子属性标记。
输出参数描述：
——pulAttribDataSize　取得的rgbAttribData参数的大小（以字节为单位）。若参数rgbAttribData为一个TSM_UNICODE字符串，则该大小还包括结束符NULL。
——prgbAttribData　　此参数指向一个存放获取的属性值的缓冲区。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.1.5 Tspi_ChangeAuth

功能描述：
改变实体（对象）的授权数据（秘密）并将该对象指派给策略对象。所有使用授权数据的类都提供本函数用以改变它们的授权数据。
接口定义：

```
TSM_RESULT Tspi_ChangeAuth
(
    TSM_HOBJECT         hObjectToChange,    // in
    TSM_HOBJECT         hParentObject,      // in
    TSM_HPOLICY         hNewPolicy          // in
);
```

输入参数描述：
——hObjectToChange　　需要修改授权数据的对象句柄。
——hParentObject　　需要修改授权对象的父对象句柄。
——hNewPolicy　　更新的授权信息策略对象句柄。
输出参数描述：
——无。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INTERNAL_ERROR

6.3.1.6 Tspi_GetPolicyObject

功能描述：
返回当前工作对象的策略。
如果应用程序没有创建策略对象，且在调用前没有为该工作对象指派策略，本函数将返回默认的策略。为默认策略设置新的授权数据信息，将影响与其相关的所有对象的后续操作。

接口定义：

```
TSM_RESULT Tspi_GetPolicyObject
(
    TSM_HOBJECT         hObject,        // in
    TSM_FLAG            policyType,     // in
    TSM_HPOLICY*        phPolicy        // out
);
```

输入参数描述：

——hObject　　对象句柄。

——policyTypet　　定义的策略类型。

输出参数描述：

——phPolicy　　返回指派的策略对象。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.2　上下文管理

6.3.2.1 Tspi_Context_Create

功能描述：

创建一个上下文对象，返回该对象句柄。

接口定义：

```
TSM_RESULT Tspi_Context_Create
(
    TSM_HCONTEXT*   phContext       // out
);
```

输入参数描述：

——无。

输出参数描述：

——phContext　创建的上下文对象句柄。

返回参数：

TSM_SUCCESS

TSM_E_INTERNAL_ERROR

6.3.2.2 Tspi_Context_Close

功能描述：

销毁一个上下文对象，并释放所分配的所有资源。

接口定义：

```
TSM_RESULT Tspi_Context_Close
(
    TSM_HCONTEXT    hContext    // in
 );
```

输入参数描述：

——hContext　　要关闭的上下文对象的句柄。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.2.3 Tspi_SetAttribUint32（整型参数）

功能描述：

本函数设置上下文对象的32-bit 属性。如果请求的数据长度小于UINT32,必须将数据转换成32-bit长度的数据。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
    TSM_HOBJECT         hObject,        // in
    TSM_FLAG            attribFlag,     // in
    TSM_FLAG            subFlag,        // in
    UINT32              ulAttrib        // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　需要设置的属性标记。

——subFlag　　需要设置的子属性标记。

——ulAttrib　　属性设置的值。

输入参数中属性见表1。

表 1 属性说明

属性	子属性	属性值	描述
TSM_TSPATTRIB_CONTEXT_SILENT_MODE		TSM_TSPATTRIB_CONTEXT_NOT_SILENT	显示协议接口的对话窗口
		TSM_TSPATTRIB_CONTEXT_SILENT	不显示协议接口对话窗口（默认）
TSM_TSPATTRI_SECRET_HASH_MODE	TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	TSM_TSPATTRIB_HASH_MODE_NOT_NULL	对于不包括其他数据授权密码进行杂凑计算
		TSM_TSPATTRIB_HASH_MODE_NULL	对于包括其他数据授权密码进行杂凑计算

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.2.4 Tspi_GetAttribUint32（整型参数）

功能描述：

本函数获取上下文对象的32-bit 属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
    TSM_HOBJECT     hObject,    // in
    TSM_FLAG        attribFlag, // in
    TSM_FLAG        subFlag,    // in
    UINT32*         pulAttrib   // out
);
```

输入参数描述：

——hObject　　需要查询属性的对象句柄。

——attribFlag　需要查询的属性标记。

——subFlag　　需要查询的子属性标记。

输入参数中属性变量见表2。

表 2 属性说明

属性	子属性	属性值	描述
TSM_TSPATTRIB_CONTEXT_SILENT_MODE		TSM_TSPATTRIB_CONTEXT_NOT_SILENT	显示协议接口的对话窗口
		TSM_TSPATTRIB_CONTEXT_SILENT	不显示协议接口对话窗口（默认）
TSM_TSPATTRI_SECRET_HASH_MODE	TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	TSM_TSPATTRIB_HASH_MODE_NOT_NULL	对于不包括其他数据授权密码进行杂凑计算
		TSM_TSPATTRIB_HASH_MODE_NULL	对于包括其他数据授权密码进行杂凑计算

输出参数描述：

——pulAttrib　指向查询到的属性值。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.2.5 Tspi_SetAttribData(变长参数)

功能描述：

本函数设置上下文对象的非32-bit 属性。 属性数据的结构和大小依属性而定。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
    TSM_HOBJECT         hObject,            // in
    TSM_FLAG            attribFlag,         // in
    TSM_FLAG            subFlag,            // in
    UINT32              ulAttribDataSize,   // in
    BYTE*               rgbAttribData       // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。
——attribFlag　　需要设置的属性标记。
——subFlag　　需要设置的子属性标记。
——ulAttribDataSize　　prgbAttribData参数的大小（以字节为单位）
——rgbAttribData　　此参数指向一个存放设置属性值的缓冲区。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.2.6 Tspi_GetAttribData(变长参数)

功能描述：

本函数获取上下文对象的非32-bit 属性。 属性数据的结构和大小依属性而定。为属性数据所分配的内存块必须用Tspi_Context_FreeMemory来释放。

接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
    TSM_HOBJECT         hObject,            // in
```

```
    TSM_FLAG            attribFlag,          // in
    TSM_FLAG            subFlag,             // in
    UINT32*             pulAttribDataSize,   // out
    BYTE**              prgbAttribData       // out
);
```

输入参数描述：

——hObject　　需要获取属性的对象句柄。

——attribFlag　需要获取的属性标记。

——subFlag　　需要获取的子属性标记。

输出参数描述：

——pulAttribDataSize　取得的rgbAttribData参数的大小（以字节为单位）。若参数rgbAttribData为一个TSM_UNICODE字符串，则该大小还包括结束符NULL。

——prgbAttribData　　此参数指向一个存放获取的属性值的缓冲区。

输入参数的属性见表3。

表 3 属性说明

属性	子属性	属性值
TSM_TSPATTRIB_CONTEXT_MACHINE_NAME	0	提供TSM运行的机器名，是一个以NULL结束的TSM_UNICODE字符串。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.2.7 Tspi_Context_Connect

功能描述：

建立TSP与TCS的连接，TCS可以为本地的或远程的。

接口定义：

```
TSM_RESULT Tspi_Context_Connect
(
    TSM_HCONTEXT            hContext,        // in
    TSM_UNICODE*            wszDestination   // in
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——wszDestination　标识连接到的远程服务(以null结束)，NULL代表连接本地服务。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_NO_CONNECTION
TSM_E_INTERNAL_ERROR

6.3.2.8 Tspi_Context_FreeMemory

功能描述：

释放由TSP所分配的上下文内存资源。

接口定义：

```
TSM_RESULT Tspi_Context_FreeMemory
(
    TSM_HCONTEXT        hContext,   // in
    BYTE*               rgbMemory   // in
);
```

输入参数描述：

——hContext 上下文对象句柄。

——rgbMemory 上下文对象内存指针。若为NULL，则与该上下文对象绑定的内存块已经释放。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INTERNAL_ERROR
TSM_E_INVALID_RESOURCE

6.3.2.9 Tspi_Context_GetDefaultPolicy

功能描述：

获取上下文的默认策略对象。

接口定义：

```
TSM_RESULT Tspi_Context_GetDefaultPolicy
(
    TSM_HCONTEXT        hContext,   // in
    TSM_HPOLICY*        phPolicy    // out
  );
```

输入参数描述：

——hContext 上下文对象句柄。

输出参数描述：

——phPolicy 返回上下文对象的策略对象句柄。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INTERNAL_ERROR

6.3.2.10 Tspi_Context_CreateObject

功能描述：

创建并初始化一个指定类型的空对象，返回该对象的句柄。

该空对象将与一个已经打开的上下文对象相关联。

接口定义：

```
TSM_RESULT Tspi_Context_CreateObject
(
    TSM_HCONTEXT        hContext,       // in
    TSM_FLAG            objectType,     // in
    TSM_FLAG            initFlags,      // in
    TSM_HOBJECT*        phObject        // out
  );
```

输入参数描述：

——hContext　上下文对象句柄。

——objectType　要创建的对象类型标识。

——initFlags　创建的对象属性的缺省值。

输出参数描述：

——phObject　返回所创建的对象。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_OBJECT_TYPE

TSM_E_INVALID_OBJECT_INIT_FLAG

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

TSM_E_ENC_INVALID_TYPE

TSM_E_HASH_INVALID_ALG

说明：

对象类型定义见附录 A.1.2.1

对象初始化定义见附录 A.1.2.2

6.3.2.11 Tspi_Context_CloseObject

功能描述：

销毁一个上下文对象句柄所关联的对象，释放该对象相关的资源。

接口定义：

```
TSM_RESULT Tspi_Context_CloseObject
(
    TSM_HCONTEXT    hContext,   // in
    TSM_HOBJECT     hObject     // in
);
```

输入参数描述：

——hContext　上下文对象句柄。

——phObject　要释放的对象句柄。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.2.12 Tspi_Context_GetCapability

功能描述：

获取TSM应用服务或核心服务的功能/性能/属性数据。

接口定义：

```
TSM_RESULT Tspi_Context_GetCapability
(
    TSM_HCONTEXT            hContext,           // in
    TSM_FLAG                capArea,            // in
    UINT32                  ulSubCapLength,     // in
    BYTE*                   rgbSubCap,          // in
    UINT32*                 pulRespDataLength,  // out
    BYTE**                  prgbRespData        // out
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——capArea　　查询的属性名称，参见属性说明表。

——ulSubCapLength　子属性的长度参数。

——rgbSubCap　　查询的子属性参数。

输入参数属性参见表4。

表 4 属性说明

属性标记	子属性标记	数据描述
TSM_TCSCAP_ALG	TSM_ALG_XX： 代表支持的算法名称	BOOL返回：TRUE代表系统服务支持该算法，FALSE代表不支持
TSM_TCSCAP_VERSION		从系统服务获取TSM_VERSION结构说明数据
TSM_TCSCAP_CACHING	TSM_TCSCAP_PROP_KEYCACHE	BOOL返回： TRUE说明系统服务支持密钥缓存； FALSE说明不支持.
TSM_TCSCAP_CACHING	TSM_TCSCAP_PROP_AUTHCACHE	BOOL返回： TRUE说明系统服务支持授权协议缓存；FALSE 说明不支持
TSM_TCSCAP_PERSSTORAGE		BOOL返回：TRUE说明系统服务支持永久存储；FALSE 说明不支持
TSM_TSPCAP_ALG	TSM_ALG_DEFAULT	返回默认算法
TSM_TSPCAP_ALG	TSM_ALG_DEFAULT_SIZE	返回默认密钥长度
TSM_TSPCAP_ALG	TSM_ALG_XX: 代表支持的算法名称	BOOL返回:TRUE 代表支持该算法,, FALSE代表不支持

属性标记	子属性标记	数据描述
TSM_TSPCAP_VERSION		获取TSM版本
TSM_TSPCAP_PERSSTORAGE		BOOL返回：TRUE说明支持永久存储；FALSE说明不支持
TSM_TCSCAP_MANUFACTURER	TSM_TCSCAP_PROP_MANUFACTURER_ID	UINT32返回：说明系统服务厂商
	TSM_TCSCAP_PROP_MANUFACTURER_STR	返回系统服务厂商名称
TSM_TSPCAP_MANUFACTURER	TSM_TSPCAP_PROP_MANUFACTURER_ID	UINT32 返回：说明TSM厂商
	TSM_TSPCAP_PROP_MANUFACTURER_STR	返回TSM厂商名称
TSM_TSPCAP_RETURNVALUE_INFO	TSM_TSPCAP_PROP_RETURNVALUE_INFO	0说明采用ASN.1编码, 1说明采用字节流

输出参数描述：

——pulRespDataLength　返回查询的属性参数长度。

——prgbRespData　返回查询的属性数据内存地址。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.2.13 Tspi_Context_GetTCMObject

功能描述：

获取TCM对象的上下文，一个给定的上下文只对应一个TCM对象实例，代表TCM所有者。

接口定义：

```
TSM_RESULT Tspi_Context_GetTcmObject
(
    TSM_HCONTEXT        hContext,    // in
    TSM_HTCM*           phTCM        // out
);
```

输入参数描述：

——hContext 上下文对象句柄。

输出参数描述：

——phTCM 返回获取的TCM类对象句柄。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.2.14 Tspi_Context_LoadKeyByBlob

功能描述：

根据key blob中包含的信息创建密钥对象，并加载到TCM中，用hUnwrappingKey指向的密钥解密key blob。hUnwrappingKey指向的密钥必须先被加载到TCM内。

接口定义：

```
TSM_RESULT Tspi_Context_LoadKeyByBlob
(
    TSM_HCONTEXT        hContext,           // in
    TSM_HKEY            hUnwrappingKey,     // in
    UINT32              ulBlobLength,       // in
    BYTE*               rgbBlobData,        // in
    TSM_HKEY*           phKey               // out
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——hUnwrappingKey　　密钥对象句柄，此密钥用于解密包 rgbBlobData 中的密钥信息。

——ulBlobLength　　指向的密钥数据块的长度（以字节为单位）。

——rgbBlobData　　待加载的密钥数据块。

输出参数描述：

——phKey　　代表被加载密钥的Key对象句柄。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.2.15 Tspi_Context_LoadKeyByUUID

功能描述：

密钥管理器根据UUID获取密钥信息，填充密钥对象，并将密钥加载到TCM中。加载密钥时所需的父密钥信息可从永久性存储器获得。

使用此命令时，需要考虑父密钥的授权要求：

a）如果该密钥的所有父密钥都不需要授权，应用程序可以直接调用Tspi_Context_LoadKeyByUUID()来加载当前密钥。

b）如果有一个密钥需要授权，并且应用程序知道密钥缓存信息，则应用程序必须通过调用函数Tspi_Context_GetKeyByUUID()从密钥库中获得密钥信息，并为其指派策略后，用Tspi_Key_LoadKey()加载密钥。

c）如果有一个密钥需要授权，并且应用程序不知道密钥缓存信息，则应用程序必须通过调用Tspi_Context_GetRegisteredKeysByUUID()从密钥库中获取密钥缓存信息。然后为其指派策略，用Tspi_Key_LoadKey()加载密钥。

接口定义：

```
TSM_RESULT Tspi_Context_LoadKeyByUUID
  (
    TSM_HCONTEXT        hContext,               // in
    TSM_FLAG            persistentStorageType,  // in
    TSM_UUID            uuidData,               // in
    TSM_HKEY*           phKey                   // out
  );
```

输入参数描述：

——hContext　　上下文对象句柄。
——persistentStorageType　　标志，指明要加载的密钥注册在 TCS 或是 TSP 中。
——uuidData　　密钥的 UUID 值。

输出参数描述：

——phKey　　返回密钥对象句柄。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_PS_KEY_NOTFOUND
TSM_E_INTERNAL_ERROR

6.3.2.16 Tspi_Context_RegisterKey

功能描述：

将密钥注册到TSP永久性存储数据库中。

接口定义：

```
TSM_RESULT Tspi_Context_RegisterKey
(
    TSM_HCONTEXT        hContext,                   // in
    TSM_HKEY            hKey,                       // in
    TSM_FLAG            persistentStorageType,      // in
    TSM_UUID            uuidKey,                    // in
    TSM_FLAG            persistentStorageTypeParent,// in
    TSM_UUID            uuidParentKey               // out
);
```

输入参数描述：

——hContext　　上下文对象句柄。
——hKey　　待注册的密钥对象句柄。在调用本方法之前应调用 Tspi_SetAttribData()方法填充密钥对象。
——persistentStorageType　　永久存储类型，用来指明要把密钥注册到 TCS 还是 TSP 中。
——uuidKey　　分配给密钥的 UUID。
——persistentStorageTypeParent 用来指明父密钥注册在 TCS 还是 TSP 中的标志。

输出参数描述：

——uuidParentKey　父密钥的 UUID

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_PS_KEY_NOTFOUND
TSM_E_INTERNAL_ERROR

6.3.2.17 Tspi_Context_UnregisterKey

功能描述：

将密钥从TSP永久性存储数据库中注销。

接口定义：

```
TSM_RESULT Tspi_Context_UnregisterKey
(
  TSM_HCONTEXT      hContext,               // in
  TSM_FLAG          persistentStorageType,  // in
  TSM_UUID          uuidKey,                // in
  TSM_HKEY*         phkey,                  //out
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——persistentStorageType　　指明密钥注册位置（TCS 或 TSP）的标志。

——uuidKey　　密钥的 UUID。

输出参数描述：

——phKey　　返回密钥对象句柄。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_PS_KEY_NOTFOUND

TSM_E_INTERNAL_ERROR

TSM_E_BAD_PARAMETER

注：注销密钥时不需要验证授权信息

6.3.2.18 Tspi_Context_GetKeyByUUID

功能描述：

利用UUID在密钥库中查找注册的密钥信息，创建并初始化密钥对象，返回新创建的密钥对象的句柄。

接口定义：

```
TSM_RESULT Tspi_Context_GetKeyByUUID
(
  TSM_HCONTEXT      hContext,               // in
  TSM_FLAG          persistentStorageType,  // in
  TSM_UUID          uuidData,               // in
  TSM_HKEY*         phKey                   // out
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——persistentStorageType　　指明密钥注册位置（TCS 或 TSP）的标志。

——uuidData　　密钥的 UUID。

输出参数描述：

——phKey　　函数执行成功返回密钥对象句柄。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_PS_KEY_NOTFOUND
TSM_E_INTERNAL_ERROR

6.3.2.19 Tspi_Context_GetKeyByPublicInfo

功能描述：

利用给定的公钥信息在密钥库中查找到所对应的密钥信息，创建并初始化密钥对象。返回新创建的密钥对象的句柄。

接口定义：

```
TSM_RESULT Tspi_Context_GetKeyByPublicInfo
(
    TSM_HCONTEXT        hContext,               // in
    TSM_FLAG            persistentStorageType,  // in
    TSM_ALGORITHM_ID    algId,                  // in
    UINT32              ulPublicInfoLength,     // in
    BYTE*               rgbPublicInfo,          // in
    TSM_HKEY*           phKey                   // out
);
```

输入参数描述：

——hContext　　上下文对象句柄。
——persistentStorageType　　指明密钥注册位置的标志。
——algId　　所查找的密钥的算法标识。
——ulPublicInfoLength　　rgbPublicInfo 参数的长度（以字节为单位）。
——rgbPublicInfo　　公钥数据块。

输出参数描述：

——phKey　　函数执行成功返回密钥对象的接口指针。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_PS_KEY_NOTFOUND
TSM_E_INTERNAL_ERROR

6.3.2.20 Tspi_Context_GetRegisteredKeysByUUID

功能描述：

获取一个TSM_KM_KEYINFO结构的数组，该数组反映注册密钥的层次信息。

注：调用方需要调用Tspi_Context_FreeMemory()方法来释放本函数分配的内存。

接口定义：

```
TSM_RESULT Tspi_Context_GetRegisteredKeysByUUID
(
    TSM_HCONTEXT    hContext,                   // in
    TSM_FLAG        persistentStorageType,      // in
```

```
    TSM_UUID*        pUuidData,                  // in
    UINT32*          pulKeyHierarchySize,        // out
    TSM_KM_KEYINFO** ppKeyHierarchy              // out
);
```

输入参数描述：

——hContext	上下文对象句柄。
——persistentStorageType	指明密钥注册位置（TCS 或 TSP）的标志。
——pUuidDat	密钥注册后分配的 UUID。如果此参数为 NULL，返回的数据包括从根密钥开始的所有密钥的层次结构。如果此参数指定了 UUID，返回的数据包括从指定密钥开始到 SMK 的路径的层次结构，返回数据的第一个元素是指定密钥的信息，依次是其父密钥的信息，最后一个为 SMK 的信息。

输出参数描述：

——pulKeyHierarchySize	ppKeyHierarchy 参数的大小（以字节为单位）。
——ppKeyHierarchy	成功执行后，ppKeyHierarchy 指向一个存放密钥层次结构数据的缓冲区。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.2.21 Tspi_Context_SetTransEncryptionKey

功能描述：

为上下文对象设置具体的传输会话加密密钥。

接口定义：

```
TSM_RESULT Tspi_Context_SetTransEncryptionKey
(
    TSM_HCONTEXT        hContext,    // in
    TSM_HKEY            hKey         // in
);
```

输入参数描述：

——hContext	上下文对象句柄
——hKey	指定进行会话加密使用的密钥柄

输出参数描述：

——无

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.2.22 Tspi_Context_CloseTransport

功能描述：

该命令结束传输会话。

接口定义：

```
TSM_RESULT Tspi_Context_CloseTransport
(
TSM_HCONTEXT        hContext,               // in
);
```

输入参数描述：

——hContext 上下文对象句柄。

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

注：传输会话接口请见下文 7.4.6 章节的详细说明

6.3.3 管理策略类

6.3.3.1 Tspi_SetAttribUint32

功能描述：

本函数设置对象的 32-bit 属性。如果请求的数据长度小于 UINT32,必须将数据转换成正确的大小。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
    TSM_HOBJECT     hObject,    // in
    TSM_FLAG        attribFlag, // in
    TSM_FLAG        subFlag,    // in
    UINT32          ulAttrib    // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄 。

——attribFlag　需要设置的属性标记 。

——subFlag　　需要设置的子属性标记 。

——ulAttrib　　属性设置的值。

输入参数属性参见表5。

表 5 属性说明

属性标识	子属性标识	数据描述
TSM_TSPATTRIB_POLICY_CALLBACK_HMAC	应用程序提供	
TSM_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP	应用程序提供	
TSM_TSPATTRIB_POLICY_SECRET_LIFETIME	TSM_TSPATTRIB_POLSECRET_LIFETI	授权数据将一直有效

属性标识	子属性标识	数据描述
	ME_ALWAYS	
	TSM_TSPATTRIB_POLSECRET_LIFETIME_COUNTER	授权数据能够被使用的次数
	TSM_TSPATTRIB_POLSECRET_LIFETIME_TIMER	授权数据能够被使用秒数
TSM_TSPATTRIB_SECRET_HASH_MODE	TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	TSM_TSPATTRIB_HASH_MODE_NULL（默认）
		TSM_TSPATTRIB_HASH_MODE_NOT_NULL

输出参数描述：

——无

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.3.2 Tspi_GetAttribUint32

功能描述：

本函数获取对象的 32-bit 属性。如果请求的数据长度小于 UINT32,必须将数据转换成正确的大小。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
    TSM_HOBJECT     hObject,     // in
    TSM_FLAG        attribFlag, // in
    TSM_FLAG        subFlag,     // in
    UINT32  *       pulAttrib    // out
);
```

输入参数描述：

——hObject　　需要获取属性的对象句柄。

——attribFlag　需要获取的属性标记。

——subFlag　　需要获取的子属性标记。

输入参数属性参见表6。

表 6 属性说明

属性标识	子属性标识	数据描述
TSM_TSPATTRIB_POLICY_CALLBACK_HMAC	应用程序提供	

属性标识	子属性标识	数据描述
TSM_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP	应用程序提供	
TSM_TSPATTRIB_POLICY_SECRET_LIFETIME	TSM_TSPATTRIB_POLSECRET_LIFETIME_ALWAYS	授权数据将一直有效
	TSM_TSPATTRIB_POLSECRET_LIFETIME_COUNTER	授权数据能够被使用的次数
	TSM_TSPATTRIB_POLSECRET_LIFETIME_TIMER	授权数据能够被使用秒数
TSM_TSPATTRIB_SECRET_HASH_MODE	TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	TSM_TSPATTRIB_HASH_MODE_NULL（默认）
		TSM_TSPATTRIB_HASH_MODE_NOT_NULL

输出参数描述：

——pulAttrib　　接收属性设置的值。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.3.3 Tspi_SetAttribData

功能描述：

本函数设置对象的非 32-bit 属性。 属性数据的结构和大小依属性而定。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
    TSM_HOBJECT     hObject,              // in
    TSM_FLAG        attribFlag,           // in
    TSM_FLAG        subFlag,              // in
    UINT32          ulAttribDataSize,     // in
    BYTE *          rgbAttribData         // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　指示要设置的属性的标志。

——subFlags　　指示要设置的属性的子标志。

输入参数属性见表 7。

表 7 属性说明

属性标识	子属性标识	数据描述
TSM_TSPATTRIB_POLICY_CALLBACK_HMAC	应用程序提供	
TSM_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP	应用程序提供	
TSM_TSPATTRIB_POLICY_POPUPSTRING	0	POPUP窗口的字符串

——ulAttribDataSize　　rgbAttribData 参数的大小（以字节为单位）。
——rgbAttribData　　此参数指向一个存放设置属性值的缓冲区。
输出参数描述：
——无。
返回参数

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.3.4 Tspi_GetAttribData

功能描述：
本函数获取对象的非32-bit 属性。 属性数据的结构和大小依属性而定。对于开辟的内存空间，需要在Tspi_Context_FreeMemory中释放。
接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
    TSM_HOBJECT hObject,         // in
    TSM_FLAG attribFlag,         // in
    TSM_FLAG subFlag,            // in
    UINT32* pulAttribDataSize,   // out
    BYTE** prgbAttribData        // out
);
```

输入参数描述：
——hObject　　需要获取属性的对象句柄。
——attribFlag　　指示要获取的属性的标志。
——subFlags　　指示要获取的属性的子标志。
输入参数属性见表 8。

表 8 属性说明

属性标识	子属性标识	数据描述
TSM_TSPATTRIB_POLICY_CALLBACK_HMAC	应用程序提供	
TSM_TSPATTRIB_POLICY_CALLBACK_TAKEOWNERSHIP	应用程序提供	
TSM_TSPATTRIB_POLICY_POPUPSTRING	0	POPUP窗口的字符串

输出参数描述：

——pulAttribDataSize　prgbAttribData 参数的大小（以字节为单位）。
——prgbAttribData　　此参数指向一个存放指定属性值的缓冲区。
返回参数
　　TSM_SUCCESS
　　TSM_E_INVALID_HANDLE
　　TSM_E_INVALID_ATTRIB_FLAG
　　TSM_E_INVALID_ATTRIB_SUBFLAG
　　TSM_E_INVALID_ATTRIB_DATA
　　TSM_E_BAD_PARAMETER
　　TSM_E_INTERNAL_ERROR

6.3.3.5 Tspi_Policy_SetSecret

功能描述：
　　本函数设置策略对象的授权数据和句柄。
　　如果secret mode不要求任何授权数据，参数ulSecretLength 被设为0，rgbSecret为 NULL.此时，若password为NULL，则子密钥也为NULL。
　　secret mode包括：
　　TSM_SECRET_MODE_NONE: 无授权要求。
　　TSM_SECRET_MODE_SM3: 对rgbsecret指向一个散列后的32字节秘密信息。
　　TSM_SECRET_MODE_PLAIN: rgbsecret指向一段明文，ulsecretLength为rgbsecret的串长。
　　TSM_SECRET_MODE_POPUP: TSP提示用户输入一串口令，其代表了一串TSM_UNICODE 字符串。该字符串必须被SCH杂凑过。
接口定义：

```
TSM_RESULT Tspi_Policy_SetSecret
(
    TSM_HPOLICY hPolicy,        // in
    TSM_FLAG secretMode,        // in
    UINT32 ulSecretLength,      // in
    BYTE* rgbSecret             // in
);
```

输入参数描述：
——hPolicy　　策略对象句柄。
——secretMode　　策略所采用的安全模式。
——ulSecretLengthrgbSecret　　参数的字节长度。
——rgbSecret　　与策略相关的秘密数据 blob。
输出参数描述：
——无。
返回参数
　　TSM_SUCCESS
　　TSM_E_INVALID_HANDLE
　　TSM_E_BAD_PARAMETER
　　TSM_E_INTERNAL_ERROR

6.3.3.6 Tspi_Policy_FlushSecret

功能描述：

用来清除缓存的策略授权信息。

接口定义：

```
TSM_RESULT Tspi_Policy_FlushSecret
(
    TSM_HPOLICY     hPolicy     // in
);
```

输入参数描述：

——hPolicy 策略对象句柄。

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.3.7 Tspi_Policy_AssignToObject

功能描述：

该函数分配策略给一个工作对象（TCM,key,encdata）,每一个工作对象利用这个策略去发起一个授权的TCM命令。每一个工作对象在创建时都有一个默认的策略，可以由该函数给工作对象邦定一个新策略，同时要把该策略加到该工作对象的策略列表里。

接口定义：

```
TSM_RESULT Tspi_Policy_AssignToObject
(
    TSM_HPOLICY         hPolicy,    // in
    TSM_HOBJECT         hObject     // in
);
```

输入参数描述：

——hPolicy 策略对象句柄。

——hObject 工作对象句柄。

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.4 TCM 管理类

6.3.4.1 Tspi_TCM_CollateIdentityRequest

功能描述：

本函数创建平台身份密钥（PIK），绑定用户身份标记信息，返回一个证书请求包。该证书供可信方来验证这个身份密钥。

只有 TCM Owner 有权创建身份密钥。

接口定义：

```
TSM_RESULT Tspi_TCM_CollateIdentityRequest
(
    TSM_HTCM                hTCM,                       // in
    TSM_HKEY                hKeySMK,                    // in
    TSM_HKEY                hCAPubKey,                  // in
    UINT32                  ulIdentityLabelLength,      // in
    BYTE*                   rgbIdentityLabelData,       // in
    TSM_HKEY                hIdentityKey,               // in
    TSM_ALGORITHM_ID        algID ,                     // in
    UINT32*                 pulTCMIdentityReqLength,    // out
    BYTE**                  prgbTCMIdentityReq          // out
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——hKeySMK　　SMK 对象句柄。

——hCAPubKey　　含有可信方公钥信息的密钥对象句柄。

——ulIdentityLabelLength　　rgbIdentityLabelData 参数的字节数。

——rgbIdentityLabelData　　指向由用户设定的代表平台身份的字符串，为 TSM_UNICODE 字符串。

——hIdentityKey　　身份密钥对象句柄。

——algID　　对称密钥算法类型，用于标识加密 PIK 证书请求信息的对称密钥算法。

输出参数描述：

——pulTCMIdentityReqLength　　接收 prgbTCMIdentityReq 的缓冲区字节大小。

——prgbTCMIdentityReq　　指向 TCM_IDENTITY_REQ 结构的数据，为发送给可信方的请求数据。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.2 Tspi_TCM_ActivateIdentity

功能描述：

本函数验证 PIK 证书的真实性，并返回解密的证书。

接口定义：

```
TSM_RESULT Tspi_TCM_ActivateIdentity
(
      TSM_HTCM          hTCM,                           // in
      TSM_HKEY          hIdentKey,                      // in
      UINT32            ulAsymCAContentsBlobLength,     // in
      BYTE*             rgbAsymCAContentsBlob,          // in
      UINT32            ulSymCAAttestationBlobLength,   // in
```

```
        BYTE*           rgbSymCAAttestationBlob,        // in
        UINT32*         pulCredentialLength,            // out
        BYTE**          prgbCredential                  // out
    );
```

输入参数描述：

——hTCM　　指向 TCM 对象的句柄。

——hIdentityKey　　身份密钥对象句柄。

——ulAsymCAContentsBlobLength　　rgbAsymCAContentsBlob 参数的大小（单位：字节）。

——rgbAsymCAContentsBlob　　指向被加密的 TCM_ASYM_CA_CONTENTS 结构数据，其从可信方处获得。

——ulSymCAAttestationBlobLength　　rgbSymCAAttestationBlob 参数的大小（单位：字节）。

——rgbSymCAAttestationBlob　　指向被加密的 TCM_SYM_CA_ATTESTATION 结构数据，其从可信方获得。

输出参数描述：

——pulCredetialLength　　prgbCredential 参数的大小（单位：字节）。

——prgbCredetial　　指向被解密的 PIK 证书数据。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.3 Tspi_TCM_CollatePekRequest

功能描述：

创建 PEK 证书和 PEK 密钥请求信息。

接口定义：

```
TSM_RESULT Tspi_TCM_CollatePekRequest
(
    TSM_HTCM                hTCM,                   // in
    TSM_HKEY                hCAPubKey,              // in
    UINT32                  ulPekLabelLength,       // in
    BYTE*                   rgbPekLabelData,        // in
    TSM_ALGORITHM_ID        algID ,                 // in
    UINT32                  ulPekParamsLength,      // in
    BYTE*                   rgbPekParams,           // in
    UINT32*                 pulTCMPekReqLength,     // out
    BYTE**                  prgbTCMPekReq           // out
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——hCAPubKey　　可信方的公钥对象句柄。

——ulPekLabelLength　　rgbPekLabelData 参数的字节数。

——rgbPekLabelData　　指向身份标示的内存区指针，其指向的内容 TSM_UNICODE 类型的字符串。

——algID　　对称密钥算法类型，用于标识加密 PEK 及其证书的请求信息的对称密钥算法。
——ulPekParamsLength　　rgbPekParams 数据长度(单位：字节)。
——rgbPekParams　　PEK 密钥参数，指向 TCM_KEY_PARMS 结构数据。
输出参数描述：
——pulTCMPekReqLength　　接收 prgbTCMPekReq 的缓冲区字节大小。
——prgbTCMPekReq　　指向用于请求 PEK 及其证书的 TCM_PEK_REQ 结构数据。
返回参数
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.4 Tspi_TCM_ActivatePEKCert

功能描述：
本函数验证 PEK 证书的真实性，并返回解密的证书。
接口定义：

```
TSM_RESULT Tspi_TCM_ActivatePEKCert
(
    TSM_HTCM            hTCM,                           // in
    UINT32              ulAsymCAContentsBlobLength,     // in
    BYTE*               rgbAsymCAContentsBlob,          // in
    UINT32              ulSymCAAttestationBlobLength,   // in
    BYTE*               rgbSymCAAttestationBlob,        // in
    UINT32*             pulCredentialLength,            // out
    BYTE**              prgbCredential                  // out
);
```

输入参数描述：
——hTCM　　指向 TCM 对象的句柄。
——ulAsymCAContentsBlobLength　　rgbAsymCAContentsBlob 参数的大小（单位：字节）。
——rgbAsymCAContentsBlob　　指向被 EK 公钥加密的 TCM_SYMMETRIC_KEY 结构数据，其从可信方处获得。
——ulSymCAAttestationBlobLength　　rgbSymCAAttestationBlob 参数的大小（单位：字节）。
——rgbSymCAAttestationBlob　　指针，指向被加密的 TCM_SYM_CA_ATTESTATION 结构，其从可信方处获得。
输出参数描述：
——pulCredetialLength　　prgbCredential 参数的大小（单位：字节）。
——prgbCredetial　　指向被解密的 PEK 证书数据。
返回参数
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

说明：证书格式请参考 GM/T 0015-2012

6.3.4.5 Tspi_TCM_ActivatePEK

功能描述：

导入 PEK 密钥。

接口定义：

```
TSM_RESULT Tspi_TCM_ActivatePEK
(
    TSM_HTCM            hTCM,                           // in
    TSM_HKEY            hKeySMK,                        // in
    TSM_HKEY            hPEKKey,                        // in, out
    TSM_HPCRS           hPEKPcr,                        // in
    UINT32              ulAsymCAContentsBlobLength,     // in
    BYTE*               rgbAsymCAContentsBlob,          // in
    UINT32              ulSymCAAttestationBlobLength,   // in
    BYTE*               rgbSymCAAttestationBlob,        // in
);
```

输入参数描述：

——hTCM	指向 TCM 对象的句柄。
——hKeySMK	SMK 对象句柄。
——hPEKKey	用于承载解密之后 PEK 密钥对象句柄。
——hPEKPcr	用于指定 PEK 平台 PCR 绑定属性，可以为空。
——ulAsymCAContentsBlobLength	rgbAsymCAContentsBlob 参数的大小（单位：字节）。
——rgbAsymCAContentsBlob	指向被加密的对称密钥,为 EK 公钥加密的 TCM_SYMMETRIC_KEY 结构数据，其从可信方处获得。
——ulSymCAAttestationBlobLength	rgbSymCAAttestationBlob 参数的大小（单位：字节）。
——rgbSymCAAttestationBlob	指向被对称密钥加密的 PEK 密钥(TCM_KEY 结构)，其从可信方处获得。

输出参数描述：

——hPEKKey	指向解密之后 PEK 密钥对象句柄。

返回参数

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.6 Tspi_TCM_CreateEndorsementKey

功能描述：

创建不可撤消的密码模块密钥。

hKey对象的属性由Tspi_SetAttribData()设置。

接口定义：

```
TSM_RESULT Tspi_TCM_CreateEndorsementKey
(
```

```
    TSM_HTCM            hTCM,               // in
    TSM_HKEY            hKey,               // in
    TSM_VALIDATION*     pValidationData     // in, out
);
```

输入参数描述：

——hTCM 指向 TCM 对象的句柄。

——hKey 指向 endorsement Key 的密钥对象句柄，该对象描述了 endorsement Key 的创建属性。

——pValidationData 提供用于计算签名的 externalData。

输出参数描述：

——pValidationData 该命令执行成功后，该缓冲区包含计算得到的验证数据。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.7 Tspi_TCM_GetPubEndorsementKey

功能描述：

本函数获取不可撤消的密码模块的公钥。

接口定义：

```
TSM_RESULT Tspi_TCM_GetPubEndorsementKey
(
    TSM_HTCM            hTCM,                   // in
    TSM_BOOL            fOwnerAuthorized,       // in
    TSM_VALIDATION*     pValidationData,        // in, out
    TSM_HKEY*           phEndorsementPubKey     // out
);
```

输入参数描述：

——hTCM 指向 TCM 对象的句柄。

——fOwnerAuthorized 如果为真，若要获取 endorsement 的公钥，TCM owner 的秘密必须提供，否则不需要。

——pValidationData 提供用于计算签名的 externalData。

输出参数描述：

——pValidationData 该命令执行成功后，pValidationData 包含计算得到的验证数据。

——phEndorsementPubKey 获取到指向 endorsement key 公钥的密钥对象句柄。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.8 Tspi_TCM_CreateRevocableEndorsementKey

功能描述：

创建一个可撤销的 TCM Endorsement 密钥对，输入输出参数描述了撤销 EK 的授权是由 TCM 实现还是应用程序实现。撤销 EK 的授权数据应由调用者来保护和管理。

接口定义：

```
TSM_RESULT Tspi_TCM_CreateRevocableEndorsementKey
(
    TSM_HTCM            hTCM,                   // in
    TSM_HKEY            hKey,                   // in
    TSM_VALIDATION*     pValidationData,        // in, out
    UINT32*             pulEkResetDataLength,   // in, out
    BYTE**              prgbEkResetData         // in, out
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——hKey　　endorsemntKey 句柄。

——pValidationData　　提供用于计算签名的 externalData。

——pulEkResetDataLength　　如果这一值为 0，TSP 使用 TCM 来产生撤销 EK 的授权数据，之后，该参数包含撤销 EK 的授权数据包的长度。如果是其他值，则表示了由外部产生的撤销 EK 的授权数据包长度。

——prgbEkResetData　　若 pulEkResetDataLength 不为 0,该数据为外部创建的撤销 EK 的授权数据。

输出参数描述：

——pValidationData　　该命令执行成功后，该缓冲区包含计算得到的验证数据。

——pulEkResetDataLength　　如果这一值为0，TSP使用TCM来产生撤销EK的授权数据，之后，该参数包含复位数据包的长度。如果是其他值，则表示了由外部产生的复位数据包长度。

——prgbEkResetData　　若输入的*pulEKResetDataLength 为 0，则此变量的输出表示由 TCM 产生的撤销 EK 授权数据。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.9 Tspi_TCM_RevokeEndorsementKey

功能描述：

该命令清除 TCM 的可撤销 EK，并执行清除 Owner 的命令。效果上而言，此命令相当于清除所有的计数器（除了基准计数器）、EK、SMK、owner 的授权和与 Owner Auth 相关的 NVRAM。该命令不涉及其他的 NVRAM。

接口定义：

```
TSM_RESULT Tspi_TCM_RevokeEndorsementKey
(
    TSM_HTCM        hTCM,               // in
    UINT32          ulEkResetDataLength,// in
    BYTE*           rgbEkResetData      // in
```

```
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——ulEkResetDataLength　撤销 EK 的授权数据大小。

——rgbEkResetData　　撤销 EK 的授权数据。

输出参数描述：

——无

返回参数

```
TSM_SUCCESS
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
```

6.3.4.10 Tspi_TCM_TakeOwnership

功能描述：

该方法取得 TCM 的所有权。这一过程是 Owner 插入一段共享秘密到 TCM 中，TCM 的 Owner 有权限执行一些特殊操作。当该命令执行时，隐含地执行了调用 Tspi_Context_RegisterKey 对 SMK 进行注册的过程。

接口定义：

```
TSM_RESULT Tspi_TCM_TakeOwnership
(
    TSM_HTCM        hTCM,               // in
    TSM_HKEY        hKeySMK,            // in
    TSM_HKEY        hEndorsementPubKey  // in
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——hKeySMK　　SMK (Storage Root Key)的密钥对象句柄。

——hEndorsementPubKey　Endorsement public key 的密钥对象句柄，该密钥用于加密 SMK 秘密和 TCM 秘密。

输出参数描述：

——无

返回参数

```
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INTERNAL_ERROR
```

6.3.4.11 Tspi_TCM_ClearOwner

功能描述：

本函数清除 TCM 的 ownership。

接口定义：

```
TSM_RESULT Tspi_TCM_ClearOwner
(
    TSM_HTCM    hTCM,       // in
```

```
    TSM_BOOL    fForcedClear // in
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——fForcedClear　　若为 FALSE，将执行需要 TCM Owner 授权的清除命令。否则，将执行需要提供物理访问授权的清除命令。

输出参数描述：

——无

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.4.12 Tspi_TCM_SetOperatorAuth

功能描述：

本函数设置 TCM 的操作者授权值。如果执行成功，hOperatorPolicy 就会赋给 hTCM 对象作为操作者授权策略。此命令中，操作者的授权是明文形式，如果 hOperatorPolicy 的 secret mode 是 TSM_SECRET_MODE_CALLBACK，TSP 就没有权限访问授权信息，并返回 TSM_E_POLICY_NO_SECRET 错误。

接口定义：

```
TSM_RESULT Tspi_TCM_SetOperatorAuth
(
  TSM_HTCM      hTCM,           // in
  TSM_HPOLICY   hOperatorPolicy // in
);
```

输入参数描述：

——hTCM　　TCM 对象句柄。

——hOperatorPolicy 策略对象句柄，该策略对象包含新授权信息。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.13 Tspi_TCM_SetStatus

功能描述：

本函数用于修改TCM的状态。

接口定义：

```
TSM_RESULT Tspi_TCM_SetStatus
(
    TSM_HTCM        hTCM,           // in
    TSM_FLAG        statusFlag,     // in
```

```
    TSM_BOOL        fTcmState       // in
);
```

输入参数描述：

——hTCM　　TCM管理对象句柄。

——statusFlag　表示待设置的状态或属性。

——fTcmState　表示待设置的状态或属性值。

输入参数的属性见表9

表 9 属性说明

属性	FTcmState 状态值	描述
TSM_TCMSTATUS_DISABLEOWNERCLEAR	忽略	永久禁止TCM所有者进行ClearOwner操作 此时，方法ClearOwner()中的fForcedClear参数将不再允许取FALSE值 这个设置需要所有者授权
TSM_TCMSTATUS_DISABLEFORCECLEAR	忽略	临时禁止TCM所有者的强制清除操作（这种禁止只在本次系统运行时有效，在下一次系统重新启动时将被取消） 此时，方法ClearOwner()中的fForcedClear参数将暂时不允许取TRUE值（直到下次系统重新启动为止）
TSM_TCMSTATUS_OWNERSETDISABLE	TSM_BOOL	fTCMState = TRUE：表示设置 TCM 的状态为 Disabled 该命令需要 TCM 所有者的授权
TSM_TCMSTATUS_PHYSICALDISABLE	TSM_BOOL	fTCMState = TRUE: 表示设置 TCM 的状态为 Disabled 该命令必须是物理现场的。
TSM_TCMSTATUS_PHYSICALSETDEACTIVATED	TSM_BOOL	fTCMState = TRUE: 表示设置 TCM 的状态为 Deactived 该命令必须是物理现场的
TSM_TCMSTATUS_SETTEMPDEACTIVATED	忽略	暂时将 TCM 的状态设置为 Deactived（直到下次系统重新启动为止）
TSM_TCMSTATUS_SETOWNERINSTALL	TSM_BOOL	fTCMState = TRUE: 表示允许使用 TakeOwnership()方法来取得 TCM 的所有者关系 这个操作需要物理现场
TSM_TCMSTATUS_DISABLEPUBEKREAD	TSM_BOOL	永久禁止在没有 TCM 所有者授权的情况下读取 EK 公钥信息的操作，即设置该属性后，必须有 TCM 所有者授权才能进行读取 EK 公钥信息的操作。设置了这个属性后，GetPubEndorsementKey()方法中的 fOwnerAuthorized 参数取 FALSE 值不可能再有效了 设置这个属性值需要所有者授权
TSM_TCMSTATUS_DISABLED	TSM_BOOL	将 TCM 设置为可用或不可用状态
TSM_TCMSTATUS_DEACTIVATED	TSM_BOOL	将 TCM 设置为激活或非激活状态

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.14 Tspi_TCM_GetStatus

功能描述：

查询TCM状态。

读取TCM状态标记需要所有者授权。

接口定义：

```
TSM_RESULT Tspi_TCM_GetStatus
(
    TSM_HTCM    hTCM,        // in
    TSM_FLAG    statusFlag,  // in
    TSM_BOOL*   pfTcmState   // out
);
```

输入参数描述：

——hTCM　　TCM管理对象句柄。

——statusFlag　表示待获取的状态或属性。

输出参数描述：

——fTcmState　表示待获取的状态或属性值。

输出属性说明见表10.

表 10 属性说明

属性	描述
TSM_TCMSTATUS_DISABLEOWNERCLEAR	*pfTCMState = TRUE: 表示TCM所有者授权清除所有者关系的操作已经被永久禁止了 ClearOwner()方法中输入参数fForcedClear如果取值为FALSE时将永远无效
TSM_TCMSTATUS_DISABLEFORCECLEAR	*pfTCMState = TRUE: 表示强制清除TCM的操作被暂时禁止了（直到系统下一次重新启动时为止） ClearOwner()的输入参数fForcedClear取值为TRUE也将暂时无意义了
TSM_TCMSTATUS_DISABLED	*pfTCMState = TRUE: 表示TCM的属性标志位 disabledfTCMState = TRUE
TSM_TCMSTATUS_SETTEMPDEACTIVATED	*pfTCMState = TRUE:.表示TCM暂时地被Deactived了
TSM_TCMSTATUS_SETOWNERINSTALL	*pfTCMState = TRUE: 表示可以使用TakeOwnership()方法来获取所有者关系
TSM_TCMSTATUS_DISABLEPUBEKREAD	*pfTCMState = TRUE: 表示永远禁止没有TCM所有者授权就可以进行读取EK公钥的操作，即读取EK公钥时必须进行TCM所有者授权 GetPubEndorsementKey()方法中的输入参数fOwnerAuthorized取FALSE值将不再有效。需要所有者授权
TSM_TCMSTATUS_PHYSPRES_LIFTIMELOCK	*pfTCMState = TRUE: 表示在TCM生存期内，TCM的

属性	描述
	physicalPresenceHWEnable 和physicalPresenceCMDEnable 标志都不允许修改
TSM_TCMSTATUS_PHYSPRES_HWENABLE	*pfTCMState = TRUE: 表示TCM物理在线的硬件信号被允许可以用做物理在线的标志
TSM_TCMSTATUS_PHYSPRES_CMDENABLE	*pfTCMState = TRUE: 表示允许使用TCM命令TSC_PhysicalPresence来表明物理在线
TSM_TCMSTATUS_CEKP_USED	*pfTCMState = TRUE: 表明EK密钥对是使用CreateEndorsementKey()方法来生成的 *pfTCMState = FALSE: 表明EK密钥对是由厂家创建的
TSM_TCMSTATUS_PHYSPRESENCE	*pfTCMState = TRUE: 表示物理在线的软件标志
TSM_TCMSTATUS_PHYSPRES_LOCK	*pfTCMState = TRUE: 表示改变物理在线的标志的操作是不允许的
TSM_TCMSTATUS_ENABLE_REVOKEEK	表明是否允许EK被重新设置
TSM_TCMSTATUS_NV_LOCK	*pfTCMState = TRUE: 表示NV授权访问是必需的

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.15 Tspi_TCM_GetCapability

功能描述：

获取TCM功能特性。有些功能特性信息必须提供所有者授权信息才能获得。

capArea和rgbSubCap在传给TCM前，TCS不需要作任何转换。接收到错误的值后，TCM将返回合适的错误信息。

接口定义：

```
TSM_RESULT Tspi_TCM_GetCapability
(
    TSM_HTCM      hTCM,               // in
    TSM_FLAG      capArea,            // in
    UINT32        ulSubCapLength,     // in
    BYTE*         rgbSubCap,          // in
    UINT32*       pulRespDataLength,  // out
    BYTE**        prgbRespData        // out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

——capArea　　指示查询属性的标志。

——ulSubCapLength　rgbSubCap参数的字节长度。

——rgbSubCap　　指示要查询的属性数据。

输出参数描述：

——pulRespDataLength　prgbRespData　参数的字节长度。

——prgbRespData　　返回的指定属性实际数据。

输出属性说明见表11。

表 11 属性说明

属性	子属性	描述
TSM_TCMCAP_ORD	命令码	返回布尔值 TRUE表示TCM支持该命令，FALSE表示TCM不支持该命令
TSM_TCMCAP_FLAG	忽略	永久性和易失性的比特标志位
TSM_TCMCAP_ALG	TSM_ALG_XX	返回布尔值（TSM算法ID的值） TRUE表示TCM支持该算法，FALSE表示TCM不支持该算法
TSM_TCMCAP_PROPERTY	TSM_TCMCAP_PROP_PCR	UINT32值 返回TCM支持的PCR寄存器个数
	TSM_TCMCAP_PROP_PCRMAP	返回TCM_PCR_ATTRIBUTES的比特标志位
	TSM_TCMCAP_PROP_MANUFACTURER	UINT32值 返回TCM厂商的标识
	TSM_TCMCAP_PROP_SLOTS 或者 TSM_TCMCAP_PROP_KEYS	UINT32值 返回TCM可以加载的256位ECC密钥的最大个数 可随时间和情况而改变
	TSM_TCMCAP_PROP_OWNER	布尔值 返回TRUE表示TCM成功地创建了一个所有者
	TSM_TCMCAP_PROP_MAXKEYS	UINT32值 返回TCM所支持的256位ECC密钥的最大个数，不含EK
	TSM_TCMCAP_PROP_AUTHSESSIONS	UINT32值 可用的授权会话的个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_MAXAUTHSESSIONS	UINT32值 返回TCM支持的可加载授权会话的最大个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_TRANSESSIONS	UINT32值 返回可用传输会话的个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_MAXTRANSESSIONS	UINT32值 返回TCM支持的可加载传输会话的最大个数
	TSM_TCMCAP_PROP_SESSIONS	UINT32值 返回会话池中可用会话的个数。会话池中

属性	子属性	描述
		的会话包括授权会话和传输会话，可随时间和情况而改变
	TSM_TCMCAP_PROP_MAXSESSIONS	UINT32值 返回TCM支持的最大会话个数,包括授权会话和传输会话
	TSM_TCMCAP_PROP_CONTEXTS	UINT32值 返回可保存的会话个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_MAXCONTEXTS	UINT32值 返回保存的会话的最大个数
	TSM_TCMCAP_PROP_COUNTERS	UINT32值 返回可用单调计数器的个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_MAXCOUNTERS	UINT32值 返回TCM_CreateCounter控制的单调计数器的最大个数
	TSM_TCMCAP_PROP_ACTIVECOUNTER	TCM_COUNT_ID 返回当前计数器的ID 若没有活动的计数器，则返回0xff..ff
	TSM_TCMCAP_PROP_MINCOUNTERINCTIME	UINT32值 表示单调计数器递增的最小时间间隔，该间隔以1/10秒为度量单位
	TSM_TCMCAP_PROP_TISTIMEOUTS	UINT32值四元数组，每个元素表示以毫秒计的超时值，顺序如下：TIMEOUT_A, TIMEOUT_B, TIMEOUT_C, TIMEOUT_D 由平台特定的接口规范决定在何处使用这些超时值
	TSM_TCMCAP_PROP_STARTUPEFFECTS	返回TCM_STARTUP_EFFECTS结构
	TSM_TCMCAP_PROP_MAXCONTEXTCOUNTDIST	UINT32值 返回上下文计数值的最大间距，至少必须为2^16-1
	TSM_TCMCAP_PROP_DURATION	返回UINT32值三元数组，每个元素依次表示如下三类命令以毫秒计的周期值： SMALL_DURATION, MEDIUM_DURATION, LONG_DURATION
	TSM_TCMCAP_PROP_MAXNVAVAILABLE	UINT32值 返回可分配的NV区域的最大个数，可随时间和情况而改变
	TSM_TCMCAP_PROP_INPUTBUFFERSIZE	UINT32值 返回TCM 输入缓冲区的大小（字节）
	TSM_TCMCAP_PROP_MAXNVWRITE	UINT32值

属性	子属性	描述
		未建立TCM所有者前发生的NV写操作次数
	TSM_TCMCAP_PROP_REVISION	字节：这是标准结构中的TCM版本号
	TSM_TCMCAP_PROP_ORD_AUDITED	表格：正被审计的命令码
	TSM_TCMCAP_PROP_LOCALITIES_AVAIL	TCM中可用的localities个数
TSM_TCMCAP_VERSION	忽略	返回一个标识TCM版本的TSM_VERSION结构
TSM_TCMCAP_VERSION_VAL	忽略	查询TCM的版本号
TSM_TCMCAP_NV_LIST	忽略	获取用来定义NV存储区的索引列表
TSM_TCMCAP_NV_INDEX	UINT32 (TCM_NV_INDEX)	获取TCM_NV_DATA_PUBLIC结构，此结构表示指定NV存储区的值
TSM_TCMCAP_MFR	忽略	获取厂商特定的TCM和TCM状态信息
TSM_TCMCAP_SYM_MODE	TCM_SYM_MODE_*type 类型的一种加密	布尔值 查询TCM是否支持特定类型的对称加密
TSM_TCMCAP_HANDLE	TCM_RT_* values值之一，说明是否应当返回密钥、授权会话、传输会话列表。	UINT32值数组（TCM句柄），返回TCM中当前已加载对象的句柄列表
TSM_TCMCAP_TRANS_ES	TSM_ES_*values值	布尔值 查询TCM是否在传输会话中支持特定的加密方案
TSM_TCMCAP_AUTH_ENCRYPT	TSM_ALGORITHM_ID values值	布尔值 查询TCM是否在授权会话的AuthData加密中支持特定的加密方案

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.16 Tspi_TCM_SelfTestFull

功能描述：

在初始化TCM函数时执行自检测。对每个TCM内部功能都执行自检。

接口定义：

```
TSM_RESULT Tspi_TCM_SelfTestFull
(
    TSM_HTCM    hTCM,    // in
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.4.17 Tspi_TCM_GetTestResult

功能描述：

返回厂商关于该自测结果的说明信息。

接口定义：

```
TSM_RESULT Tspi_TCM_GetTestResult
(
    TSM_HTCM        hTCM,                 // in
    UINT32*         pulTestResultLength,  // out
    BYTE**          prgbTestResult        // out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

输出参数描述：

——pulTestResultLength　　prgbTestResult参数的字节长度。

——prgbTestResult　　TCM制造商定义的信息。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.18 Tspi_TCM_GetRandom

功能描述：

获得由TCM产生的随机数。

接口定义：

```
TSM_RESULT Tspi_TCM_GetRandom
(
        TSM_HTCM hTCM,               // in
        UINT32 ulRandomDataLength,   // in
        BYTE** prgbRandomData        // out
  );
```

输入参数描述：

——hTCM　　TCM对象句柄。

——ulRandomDataLength　　请求的随机数长度。

输出参数描述：

——prgbTestResult　　返回随机数数据指针。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

注：产生的随机数需满足GM/T 0005-2012

6.3.4.19 Tspi_TCM_GetEvent

功能描述：

对一个给定的PCR索引和事件序号，返回一个PCR事件。

接口定义：

```
TSM_RESULT Tspi_TCM_GetEvent
(
   TSM_HTCM hTCM,                // in
   UINT32 ulPcrIndex,            // in
   UINT32 ulEventNumber,         // in
   TSM_PCR_EVENT* pPcrEvent      // out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

——ulPcrIndex　　请求的PCR索引。

——ulEventNumber　　请求的事件序号。

输出参数描述：

——pPcrEvent　　返回的PCR事件数据。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.20 Tspi_TCM_GetEvents

功能描述：

对一个给定PCR索引，返回一组指定数量的PCR事件。

本函数为请求的事件数据分配内存。使用Tspi_Context_FreeMemory释放该内存。

接口定义：

```
TSM_RESULT Tspi_TCM_GetEvents
(
  TSM_HTCM              hTCM,              // in
  UINT32                ulPcrIndex,        // in
  UINT32                ulStartNumber,     // in
  UINT32*               pulEventNumber,    // in, out
  TSM_PCR_EVENT**       prgPcrEvents       // out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

——ulPcrIndex　　请求的PCR索引。

——ulStartNumber　　请求的第一个事件的索引

——pulEventNumber 请求的事件个数。

输出参数描述：

——pulEventNumber 获得的prgPcrEvents参数的事件数据结构个数。

——prgPcrEvents 指向PCR事件数据数组。如果是 NULL, 只有事件的个数在pulEventNumber参数中返回。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.21 Tspi_TCM_GetEventLog

功能描述：

返回全部的事件日志。

接口定义：

```
TSM_RESULT Tspi_TCM_GetEventLog
(
  TSM_HTCM          hTCM,           // in
  UINT32*           pulEventNumber, // out
  TSM_PCR_EVENT**   prgPcrEvents    // out
);
```

输入参数描述：

——hTCM TCM对象句柄。

输出参数描述：

——pulEventNumber 获得的prgPcrEvents参数的事件数据结构个数。

——prgPcrEvents 指向PCR事件数据数组。如果是 NULL, 只有事件的个数在pulEventNumber 参数中返回。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.22 Tspi_TCM_PcrExtend

功能描述：

该函数扩展PCR寄存器并记录PCR事件日志。

接口定义：

```
TSM_RESULT Tspi_TCM_PcrExtend
(
    TSM_HTCM        hTCM,               //in
    UINT32          ulPcrIndex,             //in
    UINT32          ulPcrDataLength,    //in
```

```
    BYTE*           pbPcrData,          //in
    TSM_PCR_EVENT*  pPcrEvent,  //in
    UINT32*         pulPcrValueLength,  //out
    BYTE**          prgbPcrValue        //out
  );
```

输入参数描述：

——hTCM TCM对象句柄。

——ulPcrIndex 需要扩展的PCR索引。

——ulPcrDataLength 需要扩展的数据(参数pbPcrData)的长度。

——pbPcrData PCR扩展操作数据。如果pPcrEvent不为NULL，则该数据根据TSM_PCR_EVENT结构的rgbPcrValue参数描述，被用于杂凑计算创建。如果pPcrEvent为NULL，则该数据被直接扩展到TCM中而无需TSP处理。

——pPcrEvent 指向一个TSM_PCR_EVENT 结构，该结构包含事件入口信息。如果该指针是NULL，则没有事件入口被创建，该操作仅执行一个扩展操作。如果该指针为非NULL，则结构成员被TSP设置。

输出参数描述：

——pulPcrValueLength 参数prgbPcrValue的字节长度。

——prgbPcrValue 指向的内存块包含扩展操作后的PCR数据。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.4.23 Tspi_TCM_PcrRead

功能描述：

读一个PCR寄存器。

注：本功能为prgbPcrValue数据分配内存，该内存块必须调用Tspi_Context_FreeMemory功能释放。

接口定义：

```
TSM_RESULT Tspi_TCM_PcrRead
(
  TSM_HTCM      hTCM,               // in
  UINT32        ulPcrIndex,         // in
  UINT32*       pulPcrValueLength,  // out
  BYTE**        prgbPcrValue        // out
);
```

输入参数描述：

——hTCM TCM对象句柄。

——ulPcrIndex 要读的PCR索引号。

输出参数描述：

——pulPcrValueLength 参数prgbPcrValue的字节长度。

——prgbPcrValue 读取到的PCR数据。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.24 Tspi_TCM_PcrReset

功能描述：

重置TCM内的一个PCR寄存器的值。

注：函数能否成功执行取决于执行该命令的locality，在特定的平台中，PCR寄存器可被定义成只允许某些locality才能重置某些PCR。但PCR 16是一个例外，它总是可重置的，以便允许软件测试。

接口定义：

```
TSM_RESULT Tspi_TCM_PcrReset
(
    TSM_HTCM    hTCM,           // in
    TSM_HPCRS   hPcrComposite   // in
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

——hPcrComposite　PcrComposite对象句柄，必须是TCM_PCR_INFO结构，包含将被重置的PCR。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_WRONG_LOCALITY
TSM_E_INTERNAL_ERROR

6.3.4.25 Tspi_TCM_Quote

功能描述：

该函数调用TCM_Quote命令，提供当前平台的完整配置信息。

接口定义：

```
TSM_RESULT Tspi_TCM_Quote
(
    TSM_HTCM            hTCM,           // in
    TSM_HKEY            hIdentKey,      // in
    TSM_BOOL            fAddVersion,    // in
    TSM_HPCRS           hPcrComposite,  // in
    TSM_VALIDATION*     pValidationData, // in, out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

——hIdentKey　　签名密钥对象句柄。

——fAddVersion　如果TRUE，将TCM版本增加到输出数据；如果FALSE，TCM版本数据不会增加到输出数据中。
——hPcrComposite　PCRcomposite对象句柄，必须是TSM_PCRS_STRUCT_INFO_LONG结构类型。
——pValidationData 验证数据结构。输入时提供externalData，计算签名时使用。

输出参数描述：
——pValidationData 验证数据结构。输出时若成功执行该命令，则提供一个内存区，容纳验证数据和签名数据。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.26 Tspi_TCM_ReadCounter

功能描述：
读取当前活动的单调计数器的值。
接口定义：

```
TSM_RESULT Tspi_TCM_ReadCounter
(
    TSM_HTCM        hTCM,           // in
    UINT32 *        counterValue    // out
  );
```

输入参数描述：
——hTCM　TCM对象句柄。
输出参数描述：
——counterValue　单调计数器的当前值。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.27 Tspi_TCM_ReadCurrentTicks

功能描述：
读取 TCM 内的当前时钟计数。
接口定义：

```
TSM_RESULT Tspi_TCM_ReadCurrentTicks
(
    TSM_HTCM            hTCM,       //in
    TCM_CURRENT_TICKS*  tickCount   // out
);
```

输入参数描述：

——hTCM　　TCM对象句柄。

输出参数描述：

——tickCount　时钟计数。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.4.28 Tspi_TCM_GetAuditDigest

功能描述：

返回当前日志的摘要，一般而言，此值对每个 TCM 是唯一的。

若摘要被签名，则返回当前摘要、当前审计计数器值、被审计命令码的杂凑值、以及签名；若摘要未作签名，则返回当前摘要、当前审计计数器值、被审计命令码。

接口定义：

```
TSM_RESULT Tspi_TCM_GetAuditDigest
(
    TSM_HTCM              hTCM,           // in
    TSM_HKEY              hKey,           // in
    TSM_BOOL              closeAudit,     // in
    TCM_DIGEST*           pAuditDigest,   // out
    TCM_COUNTER_VALUE*    pCounterValue,  // out
    TSM_VALIDATION*       pValidationData,// out
    UINT32*               ordSize,        // out
    UINT32**              ordList         // out
  );
```

输入参数描述：

——hTCM　　TCM对象句柄。

——hKey　　执行签名的密钥的句柄。若为NULL，则被调用的TCM函数为TCM_GetAuditDigest，若为非NULL，则为对审计摘要进行签名的密钥的句柄，此时调用TCM函数为TCM_GetAuditDigestSigned。

——closeAudit 此标志用于表示是否在签名之后结束当前审计会话。只用于hKey为非NULL的情况。

输出参数描述：

——pAuditDigest　TCM内的审计摘要值。

——pCounterValue　审计单调计数器的当前值。

——pValidationData 验证数据。当hKey为NULL时，此参数被忽略。当hKey为非NULL时，为验证签名所必要的信息。签名操作时，其输入的ExternalData字段应为antiReplay nonce，签名操作执行成功后，pValidationData->Data字段包含被签名的数据（这是一个字节数列，用TCM_SIGN_INFO结构编码，TCM_SIGN_INFO数据字段被设置成（auditDigest || counterValue || auditedOrdinalDigest）），并且TSM_pValidationData->ValidationData被设置成由TCM返回的签名。

——ordSize　　ordList中被审计的命令个数。只在hKey为NULL时返回该值。

——ordList　　审计命令链表。只在hKey为NULL时返回该值。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.4.29 Tspi_TCM_SetOrdinalAuditStatus

功能描述：

设置一个 TCM 命令的审计状态。

接口定义：

```
TSM_RESULT Tspi_TCM_SetOrdinalAuditStatus
(
      TSM_HTCM          hTCM,           // in
      TCM_COMMAND_CODE  ordinalToAudit,     // in
      TSM_BOOL          auditState      // in
);
```

输入参数描述：

——hTCM　　TCM对象句柄。
——ordinalToAudit　需要审计的TCM命令码。
——auditState　　命令的审计状态：TRUE，该命令需要审计；FALSE，命令不需要审计。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5 密钥管理类

6.3.5.1 Tspi_ChangeAuth

功能描述：

改变实体（对象）的授权数据（秘密）并将该对象指派给策略对象。所有使用秘密的类都提供本函数用以改变它们的授权数据。

接口定义：

```
TSM_RESULT Tspi_ChangeAuth
(
    TSM_HOBJECT hObjectToChange,    // in
    TSM_HOBJECT hParentObject,      // in
    TSM_HPOLICY hNewPolicy          // in
);
```

输入参数描述：

——hObjectToChange　　需要修改授权数据的对象句柄。
——hParentObject　　需要修改授权对象的父对象句柄。

——hNewPolicy　　更新的授权信息策略对象句柄。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.5.2 Tspi_GetPolicyObject

功能描述：

返回当前工作对象的策略。如果应用程序没有创建策略对象，且在调用前没有为该工作对象指派策略，本函数将返回默认的上下文策略。为默认上下文策略设置新的授权数据信息，将影响与其相关的所有对象的后续操作。

接口定义：

```
TSM_RESULT Tspi_GetPolicyObject
(
  TSM_HOBJECT   hObject,     // in
  TSM_FLAG      policyType,  // in
  TSM_HPOLICY*  phPolicy     // out
);
```

输入参数描述：

——hObject　　对象句柄。

——policyType　　定义的策略类型。

输出参数描述：

——phPolicy　　返回指派的策略对象。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.5.3 Tspi_SetAttribUint32

功能描述：

本函数设置密钥对象的32-bit属性。如果请求的数据长度小于UINT32，必须将数据转换成正确的大小。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
    TSM_HOBJECT         hObject,        // in
    TSM_FLAG            attribFlag,     // in
    TSM_FLAG            subFlag,        // in
    UINT32              ulAttrib        // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　需要设置的属性。

——subFlag　　需要设置的子属性。

——ulAttrib　　属性设置的值。

输入参数属性见表12。

表 12 属性说明

属性标记	子属性标记	属性值	描述
TSM_TSPATTRIB_KEYREGISTER	0	TSM_TSPATTRIB_KEYREGISTER_USER	密钥注册到TSP中
	0	TSM_TSPATTRIB_KEYREGISTER_SYTEM	密钥注册到TCS中
	0	TSM_TSPATTRIB_KEYREGISTER_NO	密钥没有在TSM中注册
TSM_TSPATTRIB_KEY_INFO	TSM_TSPATTRIB_KEYINFO_USAGE	TSM_KEYUSAGE_XX	TSM密钥使用值，表示密钥的使用类型 见密钥对象的属性子标记定义
	TSM_TSPATTRIB_KEYINFO_MIGRATABLE	布尔值	若为TRUE，则密钥是可迁移的
	TSM_TSPATTRIB_KEYINFO_VOLATILE	布尔值	若为TRUE，则密钥是易失性的
	TSM_TSPATTRIB_KEYINFO_AUTHDATAUSAGE	布尔值	若为TRUE，则密钥的使用需要授权
	TSM_TSPATTRIB_KEYINFO_ALOGRITHM	TSM_ALG_XX	TSM算法ID，表示密钥算法 见算法ID定义
	TSM_TSPATTRIB_KEYINFO_ENCSCHEME	TSM_KEY_ENCSCH EME_XX	TSM加密方案，见密钥加密方案定义
	TSM_TSPATTRIB_KEYINFO_SIGSCHEME	TSM_KEY_SIGSCH EME_XX	TSM签名方案见密钥签名方案定义
	TSM_TSPATTRIB_KEYINFO_SIZE		密钥位长

属性标记	子属性标记	属性值	描述
	TSM_TSPATTRIB_KEYINFO_KEYFLAGS		密钥标志信息
	TSM_TSPATTRIB_KEYINFO_AUTHUSAGE		直接设置KeyParams中的authDataUsage

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.4 Tspi_GetAttribUint32

功能描述：

本函数获取密钥对象的32-bit 属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
    TSM_HOBJECT     hObject,        // in
    TSM_FLAG        attribFlag,     // in
    TSM_FLAG        subFlag,        // in
    UINT32*         pulAttrib       // out
);
```

输入参数描述：

——hObject　　需要查询属性的对象句柄。

——attribFlag　需要查询的属性。

——subFlag　　需要查询的子属性。

输入参数属性见表 13。

表 13 属性说明

属性标记	子属性标记	属性值	描述
TSM_TSPATTRIB_KEYREGISTER	0	TSM_TSPATTRIB_KEYREGISTER_USER	密钥注册到TSP中
	0	TSM_TSPATTRIB_KEYREGISTER_SYTEM	密钥注册到TCS中
	0	TSM_TSPATTRIB_KEYREGISTE	密钥没有在TSM中注册

属性标记	子属性标记	属性值	描述
		R_NO	
TSM_TSPATTRIB_KEY_INFO	TSM_TSPATTRIB_KEYINFO_USAGE	TSM_KEYUSAGE_XX	TSM密钥使用值，表示密钥的使用类型。见密钥对象的属性子标记定义
	TSM_TSPATTRIB_KEYINFO_MIGRATABLE	布尔值	若为TRUE，则密钥是可迁移的
	TSM_TSPATTRIB_KEYINFO_VOLATILE	布尔值	若为TRUE，则密钥是易失性的
	TSM_TSPATTRIB_KEYINFO_AUTHDATAUSAGE	布尔值	若为TRUE，则密钥的使用需要授权
	TSM_TSPATTRIB_KEYINFO_ALOGRITHM	TSM_ALG_XX	TSM算法ID值，表示密钥算法 见算法ID定义。
	TSM_TSPATTRIB_KEYINFO_ENCSCHEME	TSM_KEY_ENCSCHEME_XX	TSM加密方案，见密钥加密方案定义
	TSM_TSPATTRIB_KEYINFO_SIGSCHEME	TSM_KEY_SIGSCHEME_X X	TSM签名方案 见密钥签名方案定义
	TSM_TSPATTRIB_KEYINFO_KEYFLAGS		密钥标志信息
	TSM_TSPATTRIB_KEYINFO_AUTHUSAGE		返回authDataUsage的内容
	TSM_TSPATTRIB_KEYINFO_KEYSTRUCT	TSM_KEY_STRUCT_XX	密钥结构类型 见密钥结构类型定义
	TSM_TSPATTRIB_KEYINFO_SIZE		密钥位长
TSM_TSPATTRIB_KEY_PCR	TSM_TSPATTRIB_KEYPCR_LOCALITY_ATCREATION	创建blob时的locality modifier	
	TSM_TSPATTRIB_KEYPCR_LOCALITY_ATRELEASE	使用密钥所要求的locality modifier	

输出参数描述：

——pulAttrib　指向查询到的属性值。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.5 Tspi_SetAttribData

功能描述：

设置密钥对象的非 32-bit 属性。属性数据的结构和大小依赖于属性。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
    TSM_HOBJECT hObject,         // in
    TSM_FLAG attribFlag,         // in
    TSM_FLAG subFlag,            // in
    UINT32 ulAttribDataSize,     // in
    BYTE* rgbAttribData          // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。
——attribFlag　　指示要获取的属性的标志（见属性说明表）。
——subFlags　　指示要获取的属性的子标志（见属性说明表）。
——ulAttribDataSize prgbAttribData 参数的大小（以字节为单位）。
——rgbAttribData　　此参数指向一个存放设置属性值的缓冲区。

输入参数属性见表 14。

表 14 属性说明

属性标记	属性子标记	数据描述
TSM_TSPATTRIBKEY_BLOB	TSM_TSPATTRIB_SM2KEYBLOB_BLOB	ECC 密钥数据块
	TSM_TSPATTRIB_SM2KEYBLOB_PUBLIC_KEY	ECC 密钥数据块的公钥部分
	TSM_TSPATTRIB_SM2KEYBLOB_PRIVATE_KEY	已加密的 ECC 密钥数据块的公钥部分
	TSM_TSPATTRIB_SM4KEYBLOB_BLOB	SM4 密钥数据块

返回值：

——无。

6.3.5.6 Tspi_GetAttribData

功能描述：

获取密钥对象的非 32-bit 属性。属性数据的结构和大小依赖于属性。属性信息从密钥结构 TCM_KEY 中获得，根据不同的属性参数来获取 TCM_KEY 结构中相应的属性域的值。详细域的说明参见 TCM 规范_数据结构部分。

接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
```

```
    TSM_HOBJECT     hObject,              // in
    TSM_FLAG        attribFlag,           // in
    TSM_FLAG        subFlag,              // in
    UINT32*         pulAttribDataSize,  // out
    BYTE**          prgbAttribData        // out
);
```

输入参数描述：

——hObject 需要设置属性的对象句柄。

——attribFlag 指示要获取的属性的标记。（见属性说明表 15）

——subFlags 指示要获取的子属性的标记。（见属性说明表 15）

表 15 属性说明

属性标记	子属性标记	数据描述
TSM_TSPATTRIB_KEY_BLOB	TSM_TSPATTRIB_KEYBLOB_BLOB	以密钥数据块形式返回密钥信息，返回数据 prgbAttribData 是 TCM_KEY 结构存储信息的，这个结构信息采用主机字节序保存
	TSM_TSPATTRIB_KEYBLOB_PUBLIC_KEY	ECC 密钥数据块的公钥信息；返回数据 prgbAttribData 是 TCM_STORE_PUBKEY 存储信息的，这个结构信息采用主机字节序保存。
	TSM_TSPATTRIB_KEYBLOB_PRIVATE_KEY	ECC 密钥数据块的已被加密的私钥信息；返回数据 prgbAttribData 是 TCM_STORE_PRIVKEY 存储信息的，这个结构信息采用主机字节序保存
	TSM_TSPATTRIB_SM4KEYBLOB_BLOB	SM4 密钥数据块的密钥信息；返回数据 prgbAttribData 是 TCM_STORE_SYMKEY 存储信息的，这个结构信息采用主机字节序保存
TSM_TSPATTRIB_KEY_INFO	TSM_TSPATTRIB_KEYINFO_VERSION	返回数据 prgbAttribData 以 TSM_VERSION 结构返回的版本信息
TSM_TSPATTRIB_KEY_UUID	0	包含 UUID 由密钥来赋值的 TSM_UUID 结构
TSM_TSPATTRIB_KEY_PCR	TSM_TSPATTRIB_KEYPCR_CREATION_PCR_SELECTION	创建数据块时所选择的 PCR，返回数据 prgbAttribData 是 TCM_PCR_INFO 中的 TCM_PCR_SELECTION creationPCRSelection 存储信息的，这个结构信息采用主机字节序保存
	TSM_TSPATTRIB_KEYPCR_RELEASE_PCR_SELECTION	在用到密钥时所选择的 PCR，返回数据 prgbAttribData 是 TCM_PCR_INFO 中的 TCM_PCR_SELECTION releasePCRSelection 存储信息的，这个结构信息采用主机字节序保存
	TSM_TSPATTRIB_KEYPCR_DIGEST_ATCREATION	创建 PCR selection 结构时对应的 PCR 摘要，返回数据 prgbAttribData 是 TCM_PCR_INFO 中的 TCM_COMPOSITE_HASH digestAtCreation 存储信息的，这个结构信息采用主机字节序保存
	TSM_TSPATTRIB_KEYPCR_DIGEST_ATRELEASE	释放 PCR selection 结构时对应的 PCR 摘要，返回数据 prgbAttribData 是 TCM_PCR_INFO 中的

属性标记	子属性标记	数据描述
		TCM_COMPOSITE_HASH digestAtRelease 存储信息的，这个结构信息采用主机字节序保存

输出参数描述：

——pulAttribDataSize 返回的 prgbAttribData 参数的大小（以字节为单位）。

——prgbAttribData 命令成功返回，此参数指向一个存放指定属性值的缓冲区。

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.7 Tspi_Key_LoadKey

功能描述：

将主机的密钥加载到 TCM 中，TCM 负责将密钥解密，并缓存加载在 TCM 中；只有执行了 LoadKey 加载后，才能使用密钥进行加、解密和签名等密钥服务功能。

调用逻辑：

a) 对于密钥对象通过 Tspi_SetAttribData ()设置在的密钥信息；
b) 使用该方法之前，hKey 及 hUnwrappingKey 的策略对象必须被正确设置。
c) hUnwrappingKey 所指定的这个密钥的保护密钥需要事先被加载到 TCM。
d) 当密钥被加载后，TCM 会返回这个密钥在 TCM 中的会话句柄，在使用这个密钥时，使用这个句柄来使用这个密钥（因 TCM 资源有限，核心服务模块在本地可以提供缓存机制）。这个句柄作为密钥对象的内部变量保存和使用。

接口定义：

```
TSM_RESULT Tspi_Key_LoadKey
(
    TSM_HKEY    hKey,             // in
    TSM_HKEY    hUnwrappingKey    // in
);
```

输入参数描述：

——hKey　　被载入的密钥对象的句柄。

——hUnwrappingKey　用于解开 hKey 的密钥句柄。

输出参数描述：

——无

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.8 Tspi_Key_UnloadKey

功能描述：

卸载 TCM 内的密钥。即对于指定的密钥对象中的在 TCM 加载的句柄，通知 TCM 不再使用，可以释放资源。

接口定义：

```
TSM_RESULT Tspi_Key_UnloadKey
(
    TSM_HKEY     hKey   // in
);
```

输入参数描述：

——hKey 要卸载的密钥对象句柄。

输出参数描述：

——无

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TCM_KEY_OWNER_CONTROL

6.3.5.9 Tspi_Key_GetPubKey

功能描述：

这个接口只对于 ECC 非对称密钥有效，对于已经加载的密钥通过密钥句柄获取密钥对象的公钥。

接口定义：

```
TSM_RESULT Tspi_Key_GetPubKey
(
    TSM_HKEY        hKey,               // in
    UINT32*         pulPubKeyLength,    // out
    BYTE**          prgbPubKey          // out
);
```

输入参数描述：

——hKey 密钥对象句柄。

输出参数描述：

——pulPubKeyLength 接收 prgbPubKey 参数的数据长度。（单位：字节）
——prgbPubKey 指向内存中 hKey 密钥对象的公钥。
返回数据采用 TCM_PUBKEY 结构，主机字节序存储。

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.10 Tspi_Key_CertifyKey

功能描述：

使用一个ECC非对称密钥的私钥对另一个ECC非对称密钥的公钥签名。

接口定义：

```
TSM_RESULT Tspi_Key_CertifyKey
(
    TSM_HKEY        hKey,           // in
    TSM_HKEY        hCertifyingKey, // in
    TSM_VALIDATION* pValidationData // in, out
);
```

输入参数描述：

——hKey 密钥对象句柄，其公钥部分已被签名。

——hCertifyingKey 用来对 hKey 进行签名的密钥的句柄。

——pValidationData 是一个指向 TSM_VALIDATION 结构的指针，其成员 rgbValidationData 包含着这个命令的签名数据。

输出参数描述：

——pValidationData 是一个指向 TSM_VALIDATION 结构的指针，其成员 prgbData 指向一个含有 TCM_CERTIFY_INFO 数据流的缓冲区。

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.11 Tspi_Key_CreateKey

功能描述：

在 TCM 内部创建密钥（对称或非对称）并用 hWrappingKey 指向的密钥加密。

接口定义：

```
TSM_RESULT Tspi_Key_CreateKey
(
  TSM_HKEY      hKey,           // in
  TSM_HKEY      hWrappingKey,   // in
  TSM_HPCRS     hPcrComposite   // in,
);
```

输入参数描述：

——hKey 要创建的密钥对象的句柄。

——hWrappingKey 用来加密新创建的密钥的密钥句柄。

——hPcrComposite 为 Tspi_PcrComposite 对象的句柄，如果此句柄的值不为 NULL，新创建的密钥将会被绑定到由这个对象所描述的 PCR。

输出参数描述：

——无

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER

TSM_E_KEY_NO_MIGRATION_POLICY
TSM_E_INTERNAL_ERROR

6.3.5.12 Tspi_Key_WrapKey

功能描述：

用 hWrappingKey 指向的密钥加密 hKey 指向的密钥。

如果 hWrappingKey 指向的密钥是对称密钥则需要发送到 TCM 进行加密，如果为非对称则在 TSP 层进行加密。

接口定义：

```
TSM_RESULT Tspi_Key_WrapKey
(
  TSM_HKEY      hKey,           // in
  TSM_HKEY      hWrappingKey,   // in
  TSM_HPCRS     hPcrComposite   // in
);
```

输入参数描述：

——hKey 要加密的密钥对象的句柄。

——hWrappingKey 用来加密 hKey 的密钥句柄。

——hPcrComposite 为 Tspi_PcrComposite 的对象的句柄，如果此句柄的值非 NULL，新创建的密钥将会被绑定到由这个对象所描述的 PCR。

输出参数描述：

——无

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.13 Tspi_Key_AuthorizeMigrationKey

功能描述：

此命令生成一个用于迁移密钥的授权数据，允许TCM所有者指定迁移策略，之后的密钥迁移无需TCM所有者参与。

接口定义：

```
TSM_RESULT Tspi_Key_AuthorizeMigrationKey
(
  TSM_HTCM              hTCM,                   // in
  TSM_HKEY              hMigrationKey,          // in
  TSM_MIGRATE_SCHEME    migrationScheme,        // in
  UINT32*               pulMigrationKeyAuthSize, // out
  BYTE**                ppulMigrationKeyAuth,   // out
);
```

输入参数描述：

——hTCM 对象句柄。

——hMigrationKey　　密钥对象句柄，该密钥属于迁移的目标平台，其公钥用于保护被迁移密钥。
——migrationScheme　　迁移方案标识，为 TSM_MS_MIGRATE 或者是 TSM_MS_REWRAP。
输出参数描述：
——pulMigrationKeyAuthSize　　ppulMigrationKeyAuth 的数据长度。
——ppulMigrationKeyAuth　　指向 MigrationKey 的授权数据（TCM_MIGRATIONKEYAUTH)。
返回值：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.5.14 Tspi_Key_CreateMigrationBlob

功能描述：
生成待迁移的数据块。
接口定义：

```
TSM_RESULT Tspi_Key_CreateMigrationBlob
(
    TSM_HKEY    hKeyToMigrate,          // in
    TSM_HKEY    hParentKey,             / in
    UINT32      ulmigrationKeyAuthSize, // in
    BYTE*       rgbmigrationKeyAuth,    // in
    UINT32*     pulMigratedDataSize,    // out
    BYTE**      prgbMigratedData,       // out
    UINT32*     pulEncSymKeySize,       // out
    BYTE**      prgbEncSymKey,          // out
  );
```

输入参数描述：
——hKeyToMigrate　　指向被迁移的密钥句柄。
——hParentKey　　hKeyToMigrate 指向的密钥的父密钥句柄。
——ulmigrationKeyAuthSize　　MigrationKeyAuth 的数据长度。
——rgbmigrationKeyAuth　　MigrationKey 的授权数据（TCM_MIGRATIONKEYAUTH)。
输出参数描述：
——pulMigratedDataSize　　prgbMigratedData 的数据长度。
——prgbMigratedData　　在成功执行这个命令的情况下这个参数返回迁移数据，待迁移的密钥数据，数据结构是 TCM_STORE_ASYMKEY 或 TCM_STORE_SYMKEY 加密后的数据。
——pulEncSymKeySize　　指向被加密的对称密钥的数据长度的指针。
——prgbEncSymKey　　指向对称密钥数据的指针，该对称密钥已被 Tspi_Key_AuthorizeMigrationKey()所认证的公钥加密保护。
返回值：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER

TSM_E_KEY_NO_M2IGRATION_POLICY
TSM_E_INTERNAL_ERROR
注：此方法采用数字信封方式。

6.3.5.15 Tspi_Key_ConvertMigrationBlob

功能描述：
将 Tspi_Key_CreateMigrationBlob 产生的迁移数据块转换成普通的 wrapped key。
接口定义：

```
TSM_RESULT Tspi_Key_ConvertMigrationBlob
(
    TSM_HKEY     hMEK,               // in
    TSM_HKEY    hParentKey,          // in
    TSM_HKEY    hKeyToMigrate,       // in
    UINT32      ulMigratedDataSize,  // in
    BYTE*       rgbMigratedData,     // in
    UINT32      ulEncSymKeySize,     // in
    BYTE*       rgbEncSymKey,        // in
);
```

输入参数描述：

——hMEK　　指向迁移时的保护密钥句柄，此为非对称密钥。
——hParentKey　　给由 hKeyToMigrate 指向的密钥加密的父密钥句柄。
——hKeyToMigrate　　指向被迁移的密钥对象句柄。
——ulMigratedDataSize　　由 rgbMigrationBlob 提供的迁移数据块数据的长度。
——rgbMigratedData　　由上面的函数 Tspi_Key_CreateMigrationBlob()返回的迁移数据块的数据，待迁移的密钥数据，数据结构是 TCM_STORE_ASYMKEY 或 TCM_STORE_SYMKEY 加密后的数据。。
——ulEncSymKeySize　　已加密的对称密钥的长度。
——rgbEncSymKey　　指向已加密的对称密钥（TCM_SYMMETRIC_KEY）的指针，IV 使用系统默认数值。

输出参数描述：
——无
返回值：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.6 数据加解密类

6.3.6.1 Tspi_ChangeAuth

功能描述：
改变数据对象的授权数据(秘密)，为数据对象指派策略对象。
接口定义：

```
TSM_RESULT Tspi_ChangeAuth
```

```
(
    TSM_HOBJECT     hObjectToChange,    // in
    TSM_HOBJECT     hParentObject,      // in
    TSM_HPOLICY     hNewPolicy          // in
);
```

输入参数描述：

——hObjectToChange　　需要修改授权数据的密钥对象句柄。

——hParentObject　　由 hObjectToChange 所指向的对象的父密钥对象的句柄。

——hNewPolicy　　更新的授权信息策略对象句柄。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.6.2 Tspi_GetPolicyObject

功能描述：

返回数据对象现在被赋予的策略对象。

接口定义：

```
TSM_RESULT Tspi_GetPolicyObject
(
    TSM_HOBJECT     hObject,        // in
    TSM_FLAG        policyType,     // in
    TSM_HPOLICY*    phPolicy        // out
  );
```

输入参数描述：

——hObject 对象句柄。

——policyType 定义的策略类型。

输出参数描述：

——phPolicy 返回指派的策略对象。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.6.3 Tspi_SetAttribUint32

功能描述：

数据加密类不提供该接口，直接返回错误。

返回参数：

TSM_E_NO_IMPLEMENTATION

6.3.6.4 Tspi_GetAttribUint32

功能描述：

获取数据对象的32bit属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
  TSM_HOBJECT       hObject,        // in
  TSM_FLAG          attribFlag,     // in
  TSM_FLAG          subFlag,        // in
  UINT32*           pulAttrib       // out
);
```

输入数据描述：

——hObject　　需要获取属性的对象句柄。

——attribFlag　需要获取的属性标记（见属性说明表16）。

——subFlag　　需要获取的子属性标记（见属性说明表16）。

表 16 属性说明

属性标记	属性子标识	数据描述
TSM_TSPATTRIB_ENCDATA_PCR(与TCM对应)	TSM_TSPATTRIB_ENCDATAPCR_LOCALITY_ATCREATION	在加密完成的时候获取locality的值
	TSM_TSPATTRIB_ENCDATAPCR_LOCALITY_ATRELEASE	在要进行解密的时候获取locality的值
TSM_TSPATTRIB_ENCDATA_SYMENC_MODE	TSM_TSPATTRIB_ENCDATA_SYMENC_MODE_ECB	加密采用ECB模式
	TSM_TSPATTRIB_ENCDATA_SYMENC_MODE_CBC	加密采用CBC模式
	TSM_TSPATTRIB_ENCDATA_SYMENC_MODE_OFB	加密采用OFB模式
	TSM_TSPATTRIB_ENCDATA_SYMENC_MODE_CFB	加密采用CFB模式

输出数据描述：

——pulAttrib　　接收属性设置的值

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.6.5 Tspi_SetAttribData

功能描述：

设置数据对象的非32-bit属性。属性数据的结构和大小依赖于属性。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
```

```
    TSM_HOBJECT     hObject,            // in
    TSM_FLAG        attribFlag,         // in
    TSM_FLAG        subFlag,            // in
    UINT32          ulAttribDataSize,   // in
    BYTE*           rgbAttribData       // in
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　　指示要获取的属性的标记（见属性说明表 17）。

——subFlag　　指示要获取的子属性的标记（见属性说明表 17）。

——ulAttribDataSize　　prgbAttribData 参数的大小（以字节为单位）。

——rgbAttribData　　此参数指向一个存放设置属性值的缓冲区。

表 17 属性说明

属性标记	子属性标记	数据描述
TSM_TSPATTRIB_ENCDATA_BLOB	TSM_TSPATTRIB_ENCDATABLOB_BLOB	表示已加密的数据块

输出数据描述：

——无

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.6.6 Tspi_GetAttribData

功能描述：

本函数获取对象的非 32-bit 属性。属性数据的结构和大小依属性而定。

接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
    TSM_HOBJECT     hObject,            // in
    TSM_FLAG        attribFlag,         // in
    TSM_FLAG        subFlag,            // in
    UINT32*         pulAttribDataSize,  // out
    BYTE**          prgbAttribData      // out
);
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。

——attribFlag　　指示要获取的属性的标记。（见属性说明表 18）

——subFlag　　指示要获取的子属性的标记。(见属性说明表 18)

表 18 属性说明

属性标记	子属性标记	数据描述
TSM_TSPATTRIB_ENCDATA_BLOB	TSM_TSPATTRIB_ENCDATABLOB_BLOB	已加密数据的数据块
TSM_TSPATTRIB_ENCDATA_PCR	TSM_TSPATTRIB_ENCDATAPCRLONG_CREATION_SELECTION	封装(Sealing)时,获取表征起作用的 PCR 位图
	TSM_TSPATTRIB_ENCDATAPCRLONG_RELEASE_SELECTION	解封装时，获取表征起作用的 PCR 位图
	TSM_TSPATTRIB_ENCDATAPCRLONG_DIGEST_ATCREATION	封装时，获取所选择的 PCR Composite 摘要
	TSM_TSPATTRIB_ENCDATAPCRLONG_DIGEST_ATRELEASE	解封装时，获取所选择的 PCR Composite 摘要
TSM_TSPATTRIB_ENCDATA_IV	NULL	采用 CBC 加密模式时的初始化向量

输出参数描述：

——pulAttribDataSize　　返回的 prgbAttribData 参数的大小（以字节为单位)。

——prgbAttribData　　命令成功返回，此参数指向一个存放指定属性值的缓冲区。

返回值：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA

6.3.6.7 Tspi_Data_Seal

功能描述：

用来对一个数据块进行加密。如果要对该加密的数据进行解密，则必须使用同一平台的 Tspi_Data_Unseal（）函数。

注：这里使用的加密密钥应是不可迁移密钥。

接口定义：

```
TSM_RESULT Tspi_Data_Seal
(
    TSM_HENCDATA        hEncData,       // in
    TSM_HKEY            hEncKey,        // in
    UINT32              ulDataLength,   // in
    BYTE*               rgbDataToSeal,  // in
    TSM_HPCRS           hPcrComposite   // in
);
```

输入参数描述：

——hEncData　　数据对象句柄，当命令操作成功后，则该数据对象包含被加密数据。

——hEncKey　　对数据进行加密的密钥对象句柄，该密钥是不可迁移的。

——ulDataLength　　为参数 rgbDataToSeal 的数据长度（以字节为单位）。

——rgbDataToSeal　指向含有将要被加密的数据的内存块。
——hPcrComposite　PCR Composite 对象的句柄。包含有 TCM_PCR_INFO 信息。
输出参数描述：
——无
返回值：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_ENC_INVALID_LENGTH
TSM_E_ENC_NO_DATA
TSM_E_ENC_INVALID_TYPE
TSM_E_INTERNAL_ERROR

6.3.6.8 Tspi_Data_Unseal

功能描述：
对指定的数据块进行解密。
接口定义：

```
TSM_RESULT Tspi_Data_Unseal
(
    TSM_HENCDATA    hEncData,               // in
    TSM_HKEY        hKey,                   // in
    UINT32*         pulUnsealedDataLength,  // out
    BYTE**          prgbUnsealedData        // out
  );
```

输入参数描述：
——hEncData　包含有加密数据的数据对象的句柄。
——hKey　用来给数据进行解密且地址不可迁移的密钥对象句柄。
输出参数描述：
——pulUnsealedDataLength　指向参数 prgbUnsealedData 的数据长度的指针（以字节为单位）。
——prgbUnsealedData　如果操作成功，为指向包含明文信息缓冲区的指针。
返回值：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_ENC_INVALID_LENGTH
TSM_E_ENC_NO_DATA
TSM_E_ENC_INVALID_TYPE
TSM_E_INTERNAL_ERROR

6.3.6.9 Tspi_Data_Encrypt

功能描述：
本函数对明文数据进行加密。

采用何种密码算法取决于密钥属性。

上层应用调用加密，如果是采用SM4对称加密，上层应用程序传送的数据可以不是16的倍数，各个厂家根据自己芯片的处理能力对数据进行分解处理。在最后一个数据包前的处理，每次向TCM发送16的整数倍，每次处理返回的数据需要丢掉最后一个16字节的数据，丢掉后的数据的最后一个16个字节数据作为下一个数据包的IV。最后一个数据包不丢掉数据，将所有数据拼接后，作为加密的数据。

上层应用调用加密，如果是采用ECC非对称加密，传入的明文不超过256字节。

接口定义：

```
TSM_RESULT Tspi_Data_Encrypt
(
    TSM_HENCDATA        hEncData,           // in
    TSM_HKEY            hEncKey,            // in
    TSM_BOOL            bFinal              // in
    BYTE*               rgbDataIV,          // in
    BYTE*               rgbDataToEncrypt,   // in
    UINT32              ulDataLength,       // in
);
```

输入参数描述：

——hEncData　　含有被加密数据的对象句柄。

——hEncKey　　用于加密数据的密钥对象句柄。

——bFinal　　用于标识是否是应用加密的明文最后一个分组。

——rgbDataIV　　用于对称加密使用的IV，固定16字节，可以为空。如果为空，则由TSP管理IV，默认初始IV为16字节0x00。

——rgbDataToEncrypt　　明文数据。

——ulDataLength　　明文数据字节长度。

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_ENC_INVALID_LENGTH

TSM_E_ENC_NO_DATA

TSM_E_INTERNAL_ERROR

注：SM4算法遵循GB/T 32907-2016

SM2算法遵循GB/T 32918.4-2016和GM/T 0009-2012

6.3.6.10 Tspi_Data_Decrypt

功能描述：

本函数对密文数据进行解密。

采用何种算法取决于密钥属性。

注：本函数为明文数据分配了内存空间，调用者需显式释放该空间。

上层应用调用解密，如果是采用SM4对称解密，各个厂家根据自己芯片的处理能力对数据进行分解处理。在最后一个数据包前的处理，每次向TCM发送16的整数倍，解密使用的IV是每个数据包的最

后一个16字节,并在每个数据包后增加使用对称密钥加密16个0x16后的数据,拼接起来后发送给TCM解密,TCM解密后自动删除最后16字节的0x16,返回解密后的数据;最后一个数据包不需要添加数据。将所有解密数据拼接后，作为解密的数据。

上层应用调用加密，如果是采用ECC非对称解密，不超过最大密文数据。

接口定义：

```
TSM_RESULT Tspi_Data_Decrypt
(
  TSM_HENCDATA      hEncData,           // in
  TSM_HKEY          hEncKey,            // in
  TSM_BOOL          bFinal              // in
  BYTE*             rgbDataIV,          // in
  UINT32*           ulDataLength,       // out
  BYTE**            rgbDataDecrypted    //out
);
```

输入参数描述：

——hEncData　　含有被加密数据的对象句柄。

——hEncKey　　用于解密数据的密钥对象句柄。

——bFinal　　用于标识是否是应用解密的密文最后一个分组。

——rgbDataIV　　用于解密使用的IV，固定16字节。

输出参数描述：

——ulDataLength　　明文数据字节长度。

——rgbDataEncrypted　　一旦命令成功执行，该参数指向包含解密数据的存储区。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_ENC_INVALID_LENGTH
TSM_E_ENC_NO_DATA
TSM_E_INTERNAL_ERROR

注：SM4算法遵循GM/T 0002-2012
SM2算法遵循GB/T 32918.4-2016和GM/T 0009-2012

6.3.6.11 Tspi_Data_Envelop

功能描述：

本函数对明文数据进行数字信封加密封装。

加密后的数据通过Tspi_GetAttribData来获取数字信封数据。

接口定义：

```
TSM_RESULT Tspi_Data_Envelop
(
  TSM_HENCDATA      hEncData,           // in
  TSM_HKEY          hEncKey,            // in
  BYTE*             rgbDataToEncrypt,   // in
  UINT32            ulDataLength,       // in
```

```
);
```

输入参数描述：

——hEncData　　含有被数字信封封装数据的对象句柄。

——hEncKey　　用于加密数据的密钥对象句柄。

——rgbDataToEncrypt　　明文数据。

——ulDataLength　　明文数据字节长度。

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_ENC_INVALID_LENGTH

TSM_E_ENC_NO_DATA

TSM_E_INTERNAL_ERROR

6.3.6.12 Tspi_Data_Unenvelop

功能描述：

本函数对数字信封封装数据进行解密。

注：本函数为明文数据分配了内存空间，调用者需显式释放该空间。

接口定义：

```
TSM_RESULT Tspi_Data_Unenvelop
(
  TSM_HENCDATA      hEncData,           // in
  TSM_HKEY          hEncKey,            // in
  UINT32*           ulDataLength,       // out
  BYTE**            rgbDataDecrypted    //out
);
```

输入参数描述：

——hEncData　　含有被数字信封封装数据的对象句柄。

——hEncKey　　用于解密数据的密钥对象句柄。

输出参数描述：

——ulDataLength　　明文数据字节长度。

——rgbDataEncrypted　　一旦命令成功执行，该参数指向包含解密数据的存储区。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_ENC_INVALID_LENGTH

TSM_E_ENC_NO_DATA

TSM_E_INTERNAL_ERROR

6.3.7 PCR 操作类

6.3.7.1 Tspi_PcrComposite_SetPcrLocality

功能描述：

本函数用TCM_PCR_INFO结构设置PCRComposite对象中的LocalityAtRelease值。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_SetPcrLocality
(
  TSM_HPCRS         hPcrComposite,  //in
  UINT32            LocalityValue   //in
);
```

输入参数描述：

——hPcrComposite　　PCR Composite对象句柄。

——LocalityValue　　要设置PCR Composite对象中的LocalityAtRelease值。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_OBJ_ACCESS

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

注：LocalityAtRelease 为 PCR Composite 对象中的变量

6.3.7.2 Tspi_PcrComposite_GetPcrLocality

功能描述：

本函数用TCM_PCR_INFO结构从PCRComposite对象中获取LocalityAtRelease值。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_GetPcrLocality
(
    TSM_HPCRS       hPcrComposite,  //in
    UINT32*         pLocalityValue  //out
);
```

输入参数描述：

——hPcrComposite　PCR composite对象句柄。

输出参数描述：

——pLocalityValue　返回的Locality值。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_OBJ_ACCESS

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.7.3 Tspi_PcrComposite_GetCompositeHash

功能描述：

本函数用TCM_PCR_INFO结构获取PCRComposite对象中的digestAtRelease值。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_GetCompositeHash
(
    TSM_HPCRS        hPcrComposite,     //in
    UINT32*          pLen,              //out
    BYTE**           ppbHashData        //out
);
```

输入参数描述：

——hPcrComposite PCR composite对象句柄，从中可返回杂凑摘要值。

输出参数描述：

——pLen 参数ppbHashData的字节长度。

——ppbHashData 创建或释放时的摘要数据。

返回参数：

```
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_OBJ_ACCESS
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
```

6.3.7.4 Tspi_PcrComposite_SetPcrValue

功能描述：

本函数对一个PCRcomposite对象中给定的PCR索引设置摘要值。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_SetPcrValue
(
    TSM_HPCRS       hPcrComposite,      // in
    UINT32          ulPcrIndex,         // in
    UINT32          ulPcrValueLength,   // in
    BYTE*           rgbPcrValue         // in
);
```

输入参数描述：

——hPcrComposite 要设置PCR值的PCR composite对象句柄。

——UlPcrIndex 要设置数据的PCR索引号。

——UlPcrValueLength 参数rgbPcrValue的字节长度。

——rgbPcrValue 要设置的数据。

输出参数描述：

——无。

返回参数：

```
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
```

TSM_E_INTERNAL_ERROR

6.3.7.5 Tspi_PcrComposite_GetPcrValue

功能描述：

返回在PCR composite对象中指定PCR索引的摘要值。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_GetPcrValue
(
    TSM_HPCRS       hPcrComposite,      // in
    UINT32          ulPcrIndex,         // in
    UINT32*         pulPcrValueLength,  // out
    BYTE**          prgbPcrValue        // out
);
```

输入参数描述：

——hPcrComposite　　要从中返回PCR值的PCR composite对象句柄。

——UlPcrIndex　　要读取数据的PCR索引。

输出参数描述：

——pulPcrValueLength　　参数prgbPcrValue的字节长度。

——prgbPcrValue　　指向返回的由ulPcrIndex参数指示的PCR值。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

6.3.7.6 Tspi_PcrComposite_SelectPcrIndex

功能描述：

本函数使用TCM_PCR_INFO结构在PCR composite对象中选择一个PCR索引。该函数在创建和释放PCR composite对象时使用。PCR composite对象必须由Tspi_Context_CreateObject()创建。例如：在调用Tspi_TCM_Quote之前，需用本函数来选择PCR寄存器。

接口定义：

```
TSM_RESULT Tspi_PcrComposite_SelectPcrIndex
(
    TSM_HPCRS       hPcrComposite,  //in
    UINT32          ulPcrIndex,     //in
    UINT32          Direction       //in
);
```

输入参数描述：

——hPcrComposite　　要从中返回PCR索引的PCR composite对象句柄。

——ulPcrIndex　　要选择的PCR索引号。

——Direction　　指明选择的PCR是Creation还是Release。

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.8 NV 存储管理类

6.3.8.1 Tspi_SetAttribUint32

功能描述：

本函数设置NV存储对象的32bit属性。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
      TSM_HOBJECT   hObject,        // in
      TSM_FLAG      attribFlag,     // in
      TSM_FLAG      subFlag,        // in
      UINT32        ulAttrib        // in
  );
```

输入参数描述：

——hObject　　需要设置属性的对象句柄。
——attribFlag　需要设置的属性标记。
——subFlag　　需要设置的子属性标记。
——ulAttrib　　属性设置的值。

输入参数的属性见表19

表 19　属性说明

属性	子属性	属性值	描述
TSM_TSPATTRIB_NV_INDEX	0	UINT32	与此对象关联的NV存储区的索引
TSM_TSPATTRIB_NV_PERMISSIONS	0	UINT32	权限值
TSM_TSPATTRIB_NV_DATASIZE	0	UINT32	定义的NV存储区的大小

输出参数描述：

——无。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.8.2 Tspi_GetAttribUint32

功能描述：

本函数获取NV存储对象的32-bit 属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
    TSM_HOBJECT     hObject,     // in
    TSM_FLAG        attribFlag,  // in
    TSM_FLAG        subFlag,     // in
    UINT32*         pulAttrib    // out
);
```

输入参数描述：

——hObject　　需要查询属性的对象句柄。

——attribFlag　需要查询的属性。

——subFlag　　需要查询的子属性。

输入参数的属性见表20

表 20　属性说明

属性	子属性	属性值	描述
TSM_TSPATTRIB_NV_INDEX	0	UINT32	与此对象相关联的NV存储区域的索引号
TSM_TSPATTRIB_NV_PERMISSIONS	0	UINT32	权限许可的值
TSM_TSPATTRIB_NV_DATASIZE	0	UINT32	定义的NV存储区域大小
TSM_TSPATTRIB_NV_STATE	TSM_TSPATTRIB_NVSTATE_READSTCLEAR	Boolean	每次调用TCM_Startup(ST_Clear)时设置为FALSE，在datasize为0的ReadValuexxx之后，设置为TRUE
	TSM_TSPATTRIB_NVSTATE_WRITESTCLEAR	Boolean	每次调用TCM_Startup(ST_Clear)时设置为FALSE，在datasize为0的WriteValuexxx之后，设置为TRUE
	TSM_TSPATTRIB_NVSTATE_WRITEDEFINE	Boolean	在调用TCM_NV_DefineSpace之后设置为FALSE，在datasize为0的WriteValue的成功调用之后，设置为TRUE
TSM_TSPATTRIB_NV_PCR	TSM_TSPATTRIB_NVPCR_READLOCALITYATRELEASE	BYTE	Locality掩码，用于NV区域的PCR读取限制
	TSM_TSPATTRIB_NVPCR_WRITELOCALITYATRELEASE	BYTE	Locality掩码，用于NV区域的PCR写限制

输出参数描述：

——pulAttrib　指向查询到的属性值。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.8.3 Tspi_SetAttribData

功能描述：

非易失性存储类不提供该接口直接返回错误。

返回参数：

TSM_E_NO_IMPLEMENTATION

6.3.8.4 Tspi_GetAttribData

功能描述：

获取NV存储对象的非32-bit属性。属性数据的结构和大小依赖于属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribData
(
    TSM_HOBJECT     hObject,            // in
    TSM_FLAG        attribFlag,         // in
    TSM_FLAG        subFlag,            // in
    UINT32*         pulAttribDataSize, // out
    BYTE**          prgbAttribData      // out
);
```

输入参数描述：

——hObject　　需要获取属性的对象句柄。

——attribFlag　　需要获取的属性标识。

——subFlag　　需要获取的子属性标识。

输出参数描述：

——pulAttribDataSize　取得的rgbAttribData参数的大小（以字节为单位）。若参数rgbAttribData为一个TSM_UNICODE字符串，则该大小还包括结束符NULL。

——prgbAttribData　　此参数指向一个存放获取的属性值的缓冲区。

输出参数的属性说明见表21。

表 21　属性说明

属性	子属性	属性值
TSM_TSPATTRIB_NV_PCR	TSM_TSPATTRIB_NVPCR_READPCRSELECTION	PCR选择掩码，用于NV区域的PCR读取限制
	TSM_TSPATTRIB_NVPCR_READDIGESTATRELEASE	DigestAtRelease子属性，用于NV区域PCRread限制。
	TSM_TSPATTRIB_NVPCR_WRITEPCRSELECTION	PCR选择掩码，用于NV区域的PCR读取限制

属性	子属性	属性值
	TSM_TSPATTRIB_NVPCR_WRITEDIGESTATRELEASE	DigestAtRelease子属性，用于NV区域PCRwrite限制

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.8.5 Tspi_NV_DefineSpace

功能描述：

创建指定的非易失性存储空间。

接口定义：

```
TSM_RESULT Tspi_NV_DefineSpace
(
    TSM_HNVSTORE        hNVStore,           // in
    TSM_HPCRS           hReadPcrComposite,  // in, 可以为NULL
    TSM_HPCRS           hWritePcrComposite  // in, 可以为NULL
);
```

输入参数描述：

——hNVStore　　非易失性存储对象句柄

——hReadPcrComposite

PCR对象句柄。当这个参数为NULL时，没有PCR数据与指定的非易失性存储空间绑定；当不为NULL时，新创建的非易失性存储区需要这个句柄对象描述的PCR内容来成功的执行读操作。

——hWritePcrComposite

PCR对象句柄。当这个参数为NULL时，没有PCR数据与指定的非易失性存储空间绑定；当不为NULL时，新创建的非易失性存储区需要这个句柄对象描述的PCR内容来成功的执行写操作。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_E_NV_AREA_NOT_EXIST
TCM_BAD_INDEX
TCM_AUTH_CONFLICT
TCM_AUTHFAIL
TCM_OWNERSET
TCM_BAD_DATASIZE

TCM_MAXNVWRITES
TCM_INVALID_STRUCTURE
TCM_NOWRITE

注：本函数用于定义非易失性存储空间。注意，本函数与 Tspi_NV_ReleaseSpace 调用同一个 TCM 功能命令。当两次调用这个 TCM 功能命令时，会删除定义的非易失性存储空间。所以必须检查这个定义的非易失性存储空间是否已经定义，如果已经定义，必须返回 TSM_E_NV_AREA_EXIST 错误码。

6.3.8.6 Tspi_NV_ReleaseSpace

功能描述：

释放指定的非易失性存储空间

接口定义：

```
TSM_RESULT Tspi_NV_ReleaseSpace
(
    TSM_HNVSTORE        hNVStore    // in
);
```

输入参数描述：

——hNVStore　　非易失性存储对象句柄。

输入参数描述：

——无

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_E_NV_AREA_NOT_EXIST
TCM_AREA_LOCKED

注：本函数用于释放非易失性存储空间。注意，本函数与 Tspi_NV_DefineSpace 调用同一个 TCM 功能命令。所以必须检查这个定义的非易失性存储空间是否已经定义，如果没有定义，必须返回 TSM_E_NV_AREA_NOT_EXIST 错误码。

6.3.8.7 Tspi_NV_WriteValue

功能描述：

将数据写入指定的非易失性存储区域

接口定义：

```
TSM_RESULT Tspi_NV_WriteValue
(
    TSM_HNVSTORE        hNVStore,           // in
    UINT32              offset,             // in
    UINT32              ulDataLength,       // in
    BYTE*               rgbDataToWrite      // in
);
```

输入参数描述：

——hNVStore　　非易失性存储对象句柄。

——offset　　偏移量，即在NV区域中的写入起始地址。。

——ulDataLength　rgbDataToWrite的长度。即需要写到非易失性存储区域中的长度。
——rgbDataToWrite　指向需要写入数据的缓冲区。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TCM_BAD_INDEX
TCM_MAXNVWRITES
TCM_AUTH_CONFLICT
TCM_AUTHFAIL
TCM_AREA_LOCKED
TCM_BAD_LOCALITY
TCM_BAD_PRESENCE
TCM_DISABLED_CMD
TCM_NOSPACE
TCM_NOT_FULLWRITE
TCM_WRONGPCRVALUE
注：如果有一个策略对象分配给这个对象，策略对象中的授权数据被用于这个操作的授权。如果没有策略对象分配给这个对象，执行一个非授权写操作

6.3.8.8 Tspi_NV_ReadValue

功能描述：
从指定的非易失性存储区域中读取数据
接口定义：

```
TSM_RESULT Tspi_NV_ReadValue
(
    TSM_HNVSTORE        hNVStore,       // in
    UINT32              offset,         // in
    UINT32*             pulDataLength,  // in, out
    BYTE**              prgbDataRead    // out
);
```

输入参数描述：
——hNVStore　非易失性存储对象句柄。
——offset　需要读取数据在非易失性存储区域中的偏移值。
——pulDataLength　输入的prgbDataRead的缓冲区长度。
输出参数描述：
——pulDataLength　输出的prgbDataRead的长度。
——prgbDataRead　指向返回读取数据的缓冲区。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR
TCM_BAD_INDEX
TCM_AUTH_CONFLICT
TCM_AUTHFAIL
TCM_BAD_LOCALITY
TCM_BAD_PRESENCE
TCM_DISABLED_CMD
TCM_NOSPACE
TCM_WRONGPCRVALUE

注：如果有一个策略对象分配给这个对象，策略对象中的授权数据被用于这个操作的授权。如果没有策略对象分配给这个对象，执行一个非授权读操作。

6.3.9 杂凑计算类

6.3.9.1 Tspi_SetAttribUint32

功能描述：

设置杂凑对象的32-bit属性。

接口定义：

```
TSM_RESULT Tspi_SetAttribUint32
(
  TSM_HOBJECT   hHash,              // in
  TSM_FLAG      attribFlag,         // in
  TSM_FLAG      subFlag,            // in
  UINT32        ulAttrib            // in
);
```

输入参数描述：

——hHash　　杂凑对象句柄。
——attribFlag　声明需要设置的属性标记 。
——subFlag　　声明需要设置的子属性标记 。
——ulAttrib　　设置的属性值。

输入参数的属性见表22

表 22 属性说明

属性标记	子属性标记	值	描述
TSM_TSPATTRIB_HASH_SCHEME	0	TSM_TSPATTRIB_HASHSCHEME_NORMAL（默认）	使用普通的杂凑方法进行杂凑操作 设置这个属性后，仅能执行Tspi_Hash_UpdateHashValue,可以执行多次
	0	TSM_TSPATTRIB_HASHSCHEME_GB	使用普通的杂凑方法进行杂凑操作 设置这个属性后，仅能执行Tspi_Hash_SetUserMessageData

属性标记	子属性标记	值	描述
			，可以执行多次，但会破坏以前执行的结果

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INVALID_ATTRIB_FLAG

TSM_E_INVALID_ATTRIB_SUBFLAG

TSM_E_INVALID_ATTRIB_DATA

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

注：当 Tspi_Hash_SetUserMessageData、Tspi_Hash_SetHashValue、Tspi_Hash_GetHashValue、Tspi_Hash_Sign、Tspi_Hash_VerifySignature 和 Tspi_Hash_TickStampBlob 调用后，不能使用 Tspi_SetAttribUint32(TSM_TSPATTRIB_HASH_SCHEME)再次更改属性。

6.3.9.2 Tspi_GetAttribUint32

功能描述：

获取杂凑对象的32 bit属性。

接口定义：

```
TSM_RESULT Tspi_GetAttribUint32
(
  TSM_HOBJECT   hHash,              // in
  TSM_FLAG      attribFlag,         // in
  TSM_FLAG      subFlag,            // in
  UINT32*       pulAttrib           // out
);
```

输入参数描述：

——hHash 杂凑对象句柄。

——attribFlag 声明需要设置的属性标记。

——subFlag 声明需要设置的子属性标记 。

输入参数的属性见表23。

表 23 属性说明

属性标记	子属性标记	值	描述
TSM_TSPATTRIB_HASH_SCHEME	0	TSM_TSPATTRIB_HASHSCHEME_NORMAL（默认）	使用普通的杂凑方法进行杂凑操作仅能Tspi_Hash_UpdateHashValue，可以执行多次
	0	TSM_TSPATTRIB_HASHSCHEME_GB	使用普通的杂凑方法进行杂凑操作 仅能

属性标记	子属性标记	值	描述
			Tspi_Hash_SetUserMessageData，可以执行多次，但会破坏以前执行的结果

输出参数描述：

——pulAttrib　　获取的属性值。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.9.3 Tspi_SetAttribData

功能描述：

设置数据对象的非32-bit属性。属性数据的结构和大小依赖于属性。

接口定义：

```
TSM_RESULT Tspi_SetAttribData
(
    TSM_HOBJECT     hHash,              // in
    TSM_FLAG        attribFlag,         // in
    TSM_FLAG        subFlag,            // in
    UINT32          ulAttribDataSize,   // in
    BYTE*           rgbAttribData       // in
);
```

输入参数描述：

——hHash　　需要设置属性的Hash对象句柄。

——attribFlag　声明需要设置的属性标记 。

——subFlag　　声明需要设置的子属性标记 。

输入参数属性见表24

表 24 属性说明

属性标记	子属性标记	数据描述
TSM_TSPATTRIB_ALG_IDENTIFIER	0	设置杂凑算法标识

——ulAttribDataSize　rgbAttribDat的长度

——rgbAttribData　需设置的属性值(SCH的算法标识符)。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE
TSM_E_INVALID_ATTRIB_FLAG
TSM_E_INVALID_ATTRIB_SUBFLAG
TSM_E_INVALID_ATTRIB_DATA
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR

6.3.9.4 Tspi_GetAttribData

功能描述：

杂凑类不提供该接口，直接返回错误。

返回参数：

TSM_E_NO_IMPLEMENTATION

6.3.9.5 Tspi_Hash_SetUserMessageData

功能描述：

对输入的用户信息，消息以及相关的公钥信息按照签名过程中定义的杂凑算法进行计算。

接口定义：

```
TSM_RESULT Tspi_Hash_SetUserMessageData
(
    TSM_HHASH       hHash,                  // in
    TSM_HKEY        hKey,                   // in
    UINT32          ulUserIDSize,           // in
    BYTE*           rgbUserID,              // in
    UINT32          ulMessageSize,          // in
    BYTE*           rgbMessage,             // in
);
```

输入参数描述：

——hHash　　杂凑对象句柄。
——hKey　　公钥的密钥句柄。
——ulUserIDSize　　输入的用户信息的长度。
——rgbUserID　　输入的用户信息。
——ulMessageSize　　输入的消息长度。
——rgbMessage　　输入的消息。

返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_HASH_INVALID_LENGTH
TSM_E_HASH_NO_DATA
TSM_E_INTERNAL_ERROR

注：　仅当使用了Tspi_SetAttribUint32设置TSM_TSPATTRIB_HASHSCHEME_GB属性后，才能使用Tspi_Hash_SetUserMessageData方法对用户信息与消息进行杂凑运算。当使用了Tspi_Hash_SetUserMessageData方法后，不能再使用Tspi_SetAttribUint32设置TSM_TSPATTRIB_HASHSCHEME_NORMAL属性，也不能使用

Tspi_Hash_UpdateHashValue方法。可以使用Tspi_Hash_GetHashValue、Tspi_Hash_Sign、Tspi_Hash_verifySinature和Tspi_Hash_TickStampBlob等函数。

6.3.9.6 Tspi_Hash_SetHashValue

功能描述：

本函数设置杂凑类的 HASH 值。如果杂凑对象的标志位为 TSM_HASH_OTHER，那么就要调用Tspi_SetAttribData()设置杂凑算法。

接口定义：

```
TSM_RESULT Tspi_Hash_SetHashValue
(
    TSM_HHASH hHash,                // in
    UINT32 ulHashValueLength,       // in
    BYTE* rgbHashValue              // in
);
```

输入参数描述：

——hHash　　杂凑对象句柄。

——ulHashValueLength　　参数 rgbHashValue 的长度（以字节为单位）。

——rgbHashValue　　指向 Hash 值存储空间。

输出参数描述：

——无

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_HASH_INVALID_LENGTH

TSM_E_HASH_NO_DATA

TSM_E_INTERNAL_ERROR

注：本标准的杂凑相关的接口涉及到的杂凑算法遵循GB/T 32905-2016

6.3.9.7 Tspi_Hash_GetHashValue

功能描述：

本函数返回杂凑类的 HASH 值。

接口定义：

```
TSM_RESULT Tspi_Hash_GetHashValue
(
    TSM_HHASH   hHash,              // in
    UINT32*     pulHashValueLength, // out
    BYTE**      prgbHashValue       // out
);
```

输入参数描述：

——hHash　　Hash 对象句柄

输出参数描述：

——pulHashValueLength　　参数 prgbSignature.的长度（以字节为单位）。

——prgbSignature　　指向 Hash 值存储空间。

返回参数

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_HASH_INVALID_LENGTH
TSM_E_HASH_NO_DATA
TSM_E_INTERNAL_ERROR

6.3.9.8 Tspi_Hash_UpdateHashValue

功能描述：

本函数更新杂凑对象的HASH值。

接口定义：

```
TSM_RESULT Tspi_Hash_UpdateHashValue
(
    TSM_HHASH       hHash,          // in
    UINT32          ulDataLength,   // in
    BYTE*           rgbData         // in
);
```

输入参数描述：

——hHash　　待更新的杂凑对象句柄。

——ulDataLength　　参数 rgbData 的数据长度（以字节为单位）。

——rgbData.　　指向要被更新的数据。

输出参数描述：

——无

返回参数

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_HASH_INVALID_LENGTH
TSM_E_HASH_NO_DATA
TSM_E_INTERNAL_ERROR

6.3.9.9 Tspi_Hash_Sign

功能描述：

对杂凑后的结果使用签名密钥进行签名。

接口定义：

```
TSM_RESULT Tspi_Hash_Sign
(
    TSM_HHASH       hHash,                  // in
    TSM_HKEY        hKey,                   // in
    UINT32*         pulSignatureLength,     // out
    BYTE**          prgbSignature           // out
```

);

输入参数描述：

——hHash　　需要签名的杂凑对象句柄。

——hKey　　用于签名的密钥句柄。

输出参数描述：

——pulSignatureLength　如果操作成功，为返回的签名数据的长度。

——prgbSignature　　如果操作成功，为返回的签名数据。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_HASH_INVALID_LENGTH

TSM_E_HASH_NO_DATA

TSM_E_HASH_NO_IDENTIFIER

TSM_E_INTERNAL_ERROR

注：签名的数据必须被设置在一个杂凑对象的实例中。使用Tspi_Hash_SetHashValue()、Tspi_Hash_UpdateHash()或Tspi_Hash_SetUserMessageData()可以设置或计算杂凑数据。Tspi_Hash_Sign方法为prgbSignature分配内存。释放这个内存必须使用Tspi_Context_FreeMemory方法。

杂凑算法遵循GB/T 32905-2016

签名/验证签名遵循GB/T 32918.2-2016和GM/T 0009-2012。

6.3.9.10　Tspi_Hash_VerifySignature

功能描述：

本函数用于验证一个杂凑类的签名。

接口定义：

```
TSM_RESULT Tspi_Hash_VerifySignature
(
    TSM_HHASH       hHash,              // in
    TSM_HKEY        hKey,               // in
    UINT32          ulSignatureLength,  // in
    BYTE*           rgbSignature        // in
);
```

输入参数描述：

——hHash　　待验证 Hash 对象句柄。

——hKey　　签名密钥句柄。

——ulSignatureLength　签名数据 rgbSignature 长度。

——rgbSignature　　指向签名数据。

输出参数描述：

——无

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_HASH_INVALID_LENGTH
TSM_E_HASH_NO_DATA
TSM_E_INVALID_SIGSCHEME
TSM_E_INTERNAL_ERROR

6.3.9.11 Tspi_Hash_TickStampBlob

功能描述：

该方法用于给一个杂凑类加一个时间戳。它将一个时钟计数(tickvalue)与一个 blob 相关联，用以标识该 blob 在 tickvalue 对应的时间之前就已存在。

接口定义：

```
TSM_RESULT Tspi_Hash_TickStampBlob
(
    TSM_HHASH           hHash,              // in
    TSM_HKEY            hIdentKey,          // in
    TSM_VALIDATION*     pValidationData     // in
);
```

输入参数描述：

——hHash　　指向 20 字节的 Hash blob 句柄。
——hIdentKey　　用于执行时间戳的身份密钥。
——pValidationData　　指向用来验证签名的数据。

输出参数描述：

——无

返回参数

TSM_SUCCESS
TSM_E_INVALID_TCM_HANDLE
TSM_E_INVALID_KEY_HANDLE
TSM_E_INTERNAL_ERROR

6.3.10 密钥协商类

6.3.10.1 Tspi_Exchange_CreateKeyExchange

功能描述：

与TCM创建一个密钥交换会话句柄，并返回一个临时的曲线上的点。

密钥协商双方A与B使用这个函数生成临时点分别为Ra,Rb。用户A将Ra传送给用户B，用户B将Rb传送给用户A，再使用Tspi_Exchange_GetKeyExchange进行密钥协商计算。

当已经使用Tspi_Exchange_CreateKeyExchange创建了立了密钥交换句柄，再使用Tspi_Exchange_CreateKeyExchange函数时，返回TSM_E_EXCHANGE_HANDLE_EXIST错误码，必须使用Tspi_Exchange_ReleaseExchangeSession释放这个密钥交换句柄后，可以再次使用Tspi_Exchange_CreateKeyExchange与TCM建立密钥交换句柄。

接口定义：

```
TSM_RESULT Tspi_Exchange_CreateKeyExchange
(
    TSM_HEXCHANGE   hKeyExchange,       //in
    UINT32*         pcRxSize,           //out
```

```
    BYTE**          prgbRxPoint         //out
);
```

输入参数描述：

——hKeyExchange　　密钥交换句柄。

输出参数描述：

——pcRxSize　　输出prgbRxPoint的长度。

——prgbRxPoint　　返回一个临时的ECC曲线上的点，按照国标密码算法规定的未压缩编码形式的字符串。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

TSM_E_EXCHANGE_HANDLE_EXIST

注：密钥协商类接口使用的算法遵循GB/T 32918.2-2016和GB/T 32918.4-2016。

6.3.10.2　Tspi_Exchange_GetKeyExchange

功能描述：

与TCM创建一个密钥交换会话句柄，并返回一个临时的曲线上的点。一次协商会话只能做一次密钥交换。

如果并没有使用 Tspi_Exchange_CreateKeyExchange 创建密钥交换句柄，本函数必须返回TSM_E_EXCHANGE_HANDLE_NOT_EXIST 错误码。

在操作成功后，句柄表示成功协商之后的对称密钥句柄。

假定用户 A 与用户 B 进行密钥协商，当双方使用 Tspi_Exchange_GetKeyExchange 函数操作成功后，用户 A 的本地校验数据为 S1，发给对方的校验数据为 Sa；用户 B 的本地校验数据为 S2，发给对方的校验数据为 Sb。用户 A 需要比较 Sb 与 S1 是否相等；用户 B 需要比较 Sa 与 S2 是否相等。如果验证失败，双方使用生成的对称密钥进行加解密会失败。

接口定义：

```
TSM_RESULT Tspi_Exchange_GetKeyExchange
(
    TSM_HEXCHANGE           hKeyExchange,       // in
    TSM_HKEY                hPermanentKey,      // in
    TSM_EXCHANGE_TAG        cExchangeTag        // in
    UINT32                  cPointSize,         // in
    BYTE*                   rgbPoint,           // in
    UINT32                  cRaSize             // in
    BYTE*                   rgbRa,              // in
    UINT32                  cRbSize             // in
    BYTE*                   rgbRb,              // in
    UINT32                  cRxSize             // in
    BYTE*                   rgbRx,              // in
    TSM_HKEY*               phKey               // in ,out
    UINT32*                 pcSxSize            //out
```

```
    BYTE**                  prgbSxData,         // out
    UINT32*                 pcSySize            // out
    BYTE**                  prgbSyData,         // out
);
```

输入参数描述：

——hKeyExchange　　密钥交换句柄。

——hPermanentKey　　已经加载到 TCM 的本地静态密钥句柄。

——cExchangeTag　　协商标识为密钥协商的身份标识，1 代表发起方，2 代表响应方。

——phKey　　产生的密钥结构属性，用来作为生成密钥的的存储结构，不与 PCR 相绑定。

——cPointSize　　对方静态密钥公钥信息长度。

——rgbPoint　　对方静态密钥公钥信息。

——cRaSize　　本地个人信息长度。

——rgbRa　　本地个人信息。

——cRbSize　　对方个人信息长度。

——rgbRb　　对方个人信息。

——cRxSize　　对方临时密钥公钥信息长度。

——rgbRx　　对方临时密钥公钥信息。

输出参数描述：

——phKey　　协商的共享密钥。

——pcSxSize　　用于本地验证协商过程的数据长度。

——prgbSxData　　用于本地验证协商过程的数据。

——pcSySize　　提供给对方进行验证过程的数据长度。

——prgbSyData　　提供给对方进行验证过程的数据。

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

TSM_E_EXCHANGE_HANDLE_NOT_EXIST

6.3.10.3　Tspi_Exchange_ReleaseExchangeSession

功能描述：

释放与TCM建立密钥交换会话句柄。

如果并没有使用Tspi_Exchange_CreateKeyExchange创建密钥交换句柄，本函数必须返回TSM_E_EXCHANGE_HANDLE_NOT_EXIST错误码。

接口定义：

```
TSM_RESULT Tspi_Exchange_ReleaseExchangeSession
(
    TSM_HEXCHANGE       hKeyExchange,       // in
);
```

输入参数描述：

——hKeyExchange　　密钥交换句柄

输出参数描述：

——无

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

TSM_E_EXCHANGE_HANDLE_NOT_EXIST

6.3.11 回调函数类

6.3.11.1 Tspicb_CallbackTakeOwnership

功能描述：

本函数在Tspip_TCM_TakeOwnership使用时被调用，此时策略对象被设置为模式。

SMK和TCM的秘密都需要在Tspip_TCM_TakeOwnership中进行设置。在命令执行前，SMK和TCM的策略需要相应信息来获取被加密的秘密。如果SMK的授权被调用的应用程序加密，那么SMK的使用策略对象就需要有一个回调函数的注册。同样，如果TCM授权被该调用的应用程序加密，他的使用策略对象也应该有一个回调函数的注册。

接口定义：

```
TSM_RESULT Tspicb_CallbackTakeOwnership
(
    PVOID               lpAppData,      // in
    TSM_HOBJECT         hObject,        // in
    TSM_HKEY            hObjectPubKey , // in
    UINT32              ulSizeEncAuth,  // in
    BYTE*               rgbEncAuth      // out
);
```

输入参数描述：

——lpAppData　　指向应用程序提供的数据。

——hObject　　TCM对象句柄。

——hObjectPubKey　　EK公钥句柄。

——ulSizeEncAuth　　rgbEncAuthOwner and rgbEncAuthSMK的大小。

输出参数描述：

——rgbEncAuth　　被加密的授权。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_POLICY_NO_SECRET

TSM_E_INTERNAL_ERROR

TSM_E_ENC_INVALID_LENGTH

6.3.11.2 Tspicb_CollateIdentity

功能描述：

本函数在Tspi_CollateIdentity使用时被调用。其目的在于允许应用程序开发者使用自定义的对称密钥算法来加密身份请求数据包。本回调函数在调用context时被作为TCM对象属性进行注册。

接口定义：

```
TSM_RESULT Tspicb_CollateIdentity
(
    PVOID               lpAppData,                      // in
    UINT32              ulTCMPlainIdentityProofLength,  // in
    BYTE*               rgbTCMPlainIdentityProof,       // in
    TSM_ALGORITHM_ID    algID,                          // in
    UINT32              ulSessionKeyLength,             // out
    BYTE*               rgbSessionKey,                  // out
    UINT32*             pulTCMIdentityProofLength,      // out
    BYTE*               rgbTCMIdentityProof             // out
);
```

输入参数描述：

——lpAppData　　指向应用程序提供的数据

——ulTCMPlainIdentityProofLength　　明文方式的身份证明长度

——rgbTCMPlainIdentityProof　　指向身份请求数据结构 TCM_IDENTITY_PROOF，该结构为明文形式

——algID　　加密算法标识

输出参数描述：

——ulSessionKeyLength　　对称密钥长度

——rgbSessionKey　　指向对称密钥

——pulTCMIdentityProofLength　　加密后的身份证明长度

——rgbTCMIdentityProof　　指向加密后的身份证明

返回参数

```
TSM_SUCCESS
TSM_E_INVALID_ENCSCHEME
TSM_E_INTERNAL_ERROR
```

6.3.11.3 Tspicb_ActivateIdentity

功能描述：

本函数在 Tspi_ActivateIdentity 使用时被调用。其目的在于允许应用程序开发者使用自定义的对称密钥算法来解密来自第三方 CA 的证书。

接口定义：

```
TSM_RESULT Tspicb_ActivateIdentity
(
    PVOID       lpAppData                       // in
    UINT32      ulSessionKeyLength,             // in
    BYTE *      rgbSessionKey,                  // in
    UINT32      ulSymCAAttestationBlobLength    // in
    BYTE *      rgbSymCAAttestationBlob,        // in
    UINT32 *    pulCredentialLength,            // out
    BYTE *      rgbCredential                   // out
);
```

输入参数描述：

——lpAppData　　指向应用程序提供的数据。

——ulSessionKeyLength　　对称密钥的长度。

——rgbSessionKey　　指向会话密钥。

——ulSymCAAttestationBlobLength 从第三方 CA 获取的被加密的证书大小。

——rgbSymCAAttestationBlob　　指向从第三方 CA 获取的被加密的证书。

输出参数描述：

——pulCredentialLength　　被解密的证书大小。

——rgbCredential　　指向从第三方 CA 获取的证书。

返回参数

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_INTERNAL_ERROR

6.3.11.4 Tspicb_CallbackHMACAuth

功能描述：

当要求授权的 TCM 命令被调用，并且策略对象被设置为回调模式，该方法将被调用。当对授权进行改变时,该函数可能被注册到两个不同的策略对象。在改变之前，回调方式需要在当前对象的使用策略中注册。为了验证，回调方式应该在信的使用策略中注册。

当参数 ReturnOrVerify 为 TRUE,回调函数必须计算 HMAC 数据，若为 FALSE，回调函数必须验证 TCM 返回的 HMAC 数据。

如果服务提供者使用一个内部的 OSAP 会话，指针 rgbNonceEvenOSAP 和 rgbNonceOddOSAP 有效。此时共享秘密必须在 HMAC 时使用。

接口定义：

```
TSM_RESULT Tspicb_CallbackHMACAuth
(
    PVOID           lpAppData,              // in
    TSM_HOBJECT     hAuthorizedObject,      // in
    TSM_BOOL        ReturnOrVerify,         // in
    UINT32          ulPendingFunction,      // in
    TSM_BOOL        ContinueUse,            // in
    UINT32          ulSizeNonces,           // in
    BYTE*           rgbNonceEven,           // in
    BYTE*           rgbNonceOdd,            // in
    BYTE*           rgbNonceEvenOSAP,       // in
    BYTE*           rgbNonceOddOSAP,        // in
    UINT32          ulSizeDigestHmac,       // in
    BYTE*           rgbParamDigest,         // in
    BYTE*           rgbHmacData             // in, out
);
```

输入参数描述：

——lpAppData　　指向应用程序提供的数据。

——hAuthorizedObject　　授权对象句柄。

——ReturnOrVerify　　标志，表明是授权还是验证(CalculateHMACData)，其中：
TRUE：回调函数必须计算 HMAC；
FALSE：回调函数必须验证来自 TCM 的 HMAC。
——ulPendingFunction　TCM 命令的序号。
——ContinueUse　　继续使用授权会话的标志。继续计算或验证 rgbHmacData。
——ulSizeNonces　　nonces rgbNonceEven, rgbNonceOdd, rgbNonceEvenOSAP 和 rgbNonceOddOSAP 的大小 。
——rgbNonceEven　　TCM 产生的偶数序号随机数。用来计算或验证 rgbHmacData。
——rgbNonceOdd　　TSP 产生的奇数序号随机数。用来计算或验证 rgbHmacData。
——rgbNonceEvenOSAP　TCM 产生的与共享秘密相关的随机数。用来计算 OSAP 会话的共享秘密。
——rgbNonceOddOSAP　调用者 产生的与共享秘密相关的随机数。用来计算 OSAP 会话的共享秘密。
——ulSizeDigestHmac　rgbParamDigest 和 rgbHmacData 的大小。
——rgbParamDigest　　TCM 函数参数的 SCH 摘要：
若 ReturnOrVerify = TRUE, incoming 参数的摘要；.
若 ReturnOrVerify = FALSE, ingoing 参数的摘要。
——rgbHmacData　　输入或返回参数的授权摘要：
若 ReturnOrVerify = TRUE,要求处理 TCM 命令的授权摘要；
若 ReturnOrVerify = FALSE,从 TCM 返回的授权摘要。

输出参数描述：

——rgbHmacData　　输入或返回参数的授权摘要：
若 ReturnOrVerify = TRUE,要求处理 TCM 命令的授权摘要；
若 ReturnOrVerify = FALSE,从 TCM 返回的授权摘要。

返回参数

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_INTERNAL_ERROR
TSM_LAYER_TSP
TSM_E_BAD_PARAMETER;
TSM_E_BAD_PARAMETER
TSM_E_POLICY_NO_SECRET;
TSM_E_TSP_AUTHFAIL

6.3.11.5 Tspicb_CallbackSealxMask

功能描述：

该回调函数在 Sealx 或者 Unseal 操作中 mask 或者 unmask 数据。

在 Sealx 和 Unseal 中的 mask 操作是相同的，所以不设置一个参数来区分使用 Sealx 还是 Unseal。在两种情况中，rgbDataToMask 提供数据源，生成的数据赋值给 rgbMaskedData。

接口定义：

```
TSM_RESULT Tspicb_CallbackSealxMask
(
    PVOID               lpAppData,              // in
    TSM_HKEY            hKey,                   // in
    TSM_HENCDATA        hEncData,               // in
```

```
    TSM_ALGORITHM_ID    algID,                  // in
    UINT32              ulSizeNonces,           // in
    BYTE*               rgbNonceEven,           // in
    BYTE*               rgbNonceOdd,            // in
    UINT32              ulDataLength,           // in
    BYTE*               rgbDataToMask,          // in
    BYTE*               rgbMaskedData           // out
);
```

输入参数描述：

——lpAppData　　指向应用程序提供的数据。

——hKey　　Sealx 或者 Unseal 操作的密钥。

——hEncData　　如果是 Sealx 操作，数据对象为明文，如果是 UnSeal 操作，数据对象为密文

——algID　　用于 mask/unmask 数据的对称算法标识。

——ulSizeNonces　　大小，字节为单位，nonce 的 buffer。

——rgbNonceEven　　授权会话的当前事件 nonce。

——rgbNonceOdd　　授权会话的临时 nonce。

——ulDataLength　　被 mask 的数据的大小。

——rgbDataToMas　　被 mask 的数据的 buffer，长度为 ulDataLength。

输出参数描述：

——rgbMaskedData　　输出 buffer，保存被 mask 后的数据，长度为 ulDataLength。.

返回参数：

TSM_SUCCESS

TSM_E_INVALID_HANDLE

TSM_E_BAD_PARAMETER

TSM_E_INTERNAL_ERROR

7　TCM 核心服务

7.1　TCM 核心服务管理

7.1.1　上下文管理

7.1.1.1 Tcs_OpenContext

功能描述：

取得TCS的一个新的上下文句柄。上下文句柄用于分配函数资源，应用服务TSP等可能需要打开多个上下文。

接口定义：

```
TSM_RESULT Tcs_OpenContext
(
    TCS_CONTEXT_HANDLE *hContext    // out
);
```

输入参数描述：

——无。

输出参数描述：

——hContext 返回已经建立的TCS上下文句柄。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.1.1.2 Tcs_CloseContext

功能描述：

释放TCS上下文。

此功能释放分配给指定上下文以及上下文本身所占用的所有资源。

接口定义：

```
TSM_RESULT Tcs_CloseContext
(
    TCS_CONTEXT_HANDLE      hContext        // in
);
```

输入参数描述：

——hContext 待释放的TCS上下文句柄。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.1.1.3 Tcs_FreeMemory

功能描述：

释放TCS上下文分配的内存。如果pMemory等于NULL，则释放所有分配的内存块。

接口定义：

```
TSM_RESULT Tcs_FreeMemory
(
    TCS_CONTEXT_HANDLE       hContext,      // in
    BYTE*                    pMemory        // in
);
```

输入参数描述：

——hContext TCS上下文句柄。

——pMemory 待释放内存块指针。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.1.1.4 Tcs_GetCapability

功能描述：

获取TCS功能/性能属性信息。

接口定义：

```
TSM_RESULT Tcs_GetCapability
(
      TCS_CONTEXT_HANDLE        hContext,       // in
      TCM_CAPABILITY_AREA       capArea,        // in
      UINT32                    subCapSize,     // in
      BYTE*                     subCap,         // in
      UINT32*                   respSize,       // out
      BYTE**                    resp            // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——capArea　　属性参数

——subCapSize　　要获得的子属性参数长度。

——subCap　　要获得的子属性参数，功能/性能属性定义如表25。

表 25 属性说明

属性标记	子属性参数	描述
TSM_TCSCAP_ALG		查询是否支持该运算
TSM_TCSCAP_VERSION		查询当前TCS版本
TSM_TCSCAP_MANUFACTURER	TSM_TCSCAP_PROP_MANUFACTURER_ID	返回厂商或TCS开发者ID
	TSM_TCSCAP_PROP_MANUFACTURER_STR	返回TCS生产商名称字符串该串的内容取决于厂商并随TCS版本而改变
TSM_TCSCAP_CACHING		无子属性，标识同时查询是否支持密钥或者授权缓存
	TSM_TCSCAP_PROP_KEYCACHE	TSM_BOOL值 表示是否支持密钥缓存
	TSM_TCSCAP_PROP_AUTHCACHE	TSM_BOOL值 表示是否支持授权会话缓存
TSM_TCSCAP_PERSSTORAGE		是否支持永久存储
TSM_TCSCAP_PLATFORM_CLASS	TSM_TCSCAP_PROP_HOST_PLATFORM	返回一个TSM_PLATFORM_CLASS数据结构，只包含主机平台类的定义
	TSM_TCSCAP_PROP_ALL_PLATFORMS	返回TSM_PLATFORM_CLASS结构数列，列举与主机平台相关的所有平台 主机平台不能作为其中之一的平台类返回 对所列举的平台次序没有要求

输出参数描述：

——respSize　　返回功能/性能应答的长度。

——resp　　功能/性能属性应答。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.1.2 密钥管理

7.1.2.1 Tcs_RegisterKey

功能描述：

将密钥注册到TCS的密钥数据库中保存。

接口定义：

```
TSM_RESULT Tcs_RegisterKey
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_UUID                WrappingKeyUUID,    // in
    TSM_UUID                KeyUUID,            // in
    UINT32                  cKeySize,           // in
    BYTE*                   rgbKey,             // in
    UINT32                  cVendorDataSize,    // in
    BYTE*                   rgbVendorData       // in
);
```

输入参数描述：

——hContext TCS上下文句柄。

——WrappingKeyUUID 已经注册的父密钥的UUID。

——KeyUUID 要注册密钥的UUID。

——cKeySize 要注册的密钥blob的字节长度。

——rgbKey 要注册的密钥blob的字节流。

——cVendorDataSize 厂商特定的数据blob的字节长度，可能为0。

——rgbVendorData 厂商特定的数据blob，可能为NULL。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_KEY_ALREADY_REGISTERED

TCS_E_KEY_NOT_REGISTERED

TCS_E_FAIL

7.1.2.2 Tcs_UnregisterKey

功能描述：

将密钥从TCS密钥数据库中删除。

接口定义：

```
TSM_RESULT Tcs_UnregisterKey
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_UUID                KeyUUID         // in
```

```
);
```

输入参数描述：

——hContext TCS上下文句柄。

——KeyUUID 已注册密钥的UUID。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_KEY_NOT_REGISTERED

TCS_E_KEY_MISMATCH

TCS_E_INVALID_CONTEXTHANDLE

TCS_E_FAIL

7.1.2.3 Tcs_EnumRegisteredKeys

功能描述：

取得指定密钥在密钥数据库中的密钥链层级关系。如果没有指定pKeyUUID，则返回密钥数据库中所有密钥的层级关系。

接口定义：

```
TSM_RESULT Tcs_EnumRegisteredKeys
(
    TCS_CONTEXT_HANDLE      hContext,               // in
    TSM_UUID*               pKeyUUID,               // in
    UINT32*                 pcKeyHierarchySize,     // out
    TSM_KM_KEYINFO**        ppKeyHierarchy          // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——pKeyUUID　　需返回所在层域的密钥的UUID。如果为NULL，返回所有密钥层域。

输出参数描述：

——pcKeyHierarchySize　返回序列实体的数目。

——ppKeyHierarchy　　返回TSM_KM_KEYINFO结构的链表指针，包含已注册密钥所在层域的序列结构。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.1.2.4 Tcs_GetRegisteredKey

功能描述：

取得指定密钥的TSM_KM_KEYINFO密钥信息。

接口定义：

```
TSM_RESULT Tcs_GetRegisteredKey
```

```
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_UUID                KeyUUID,            // in
    TSM_KM_KEYINFO**        ppKeyInfo           // out
);
```

输入参数描述：

——hContext TCS上下文句柄。

——pKeyUUID 密钥信息的UUID。

输出参数描述：

——ppKeyInfo 指定密钥的密钥信息指针。

返回参数：

TCS_SUCCESS

TCS_E_KEY_NOT_REGISTERED

TCS_E_FAIL

7.1.2.5 Tcs_GetRegisteredKeyBlob

功能描述：

取得指定密钥的密钥数据，这个数据中包含密钥的参数，如公钥、密钥使用方案、密钥的加密签名方案、以及加密的私钥等信息。

接口定义：

```
TSM_RESULT Tcs_GetRegisteredKeyBlob
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_UUID                KeyUUID,        // in
    UINT32*                 pcKeySize,      // out
    BYTE**                  prgbKey         // out
);
```

输入参数描述：

——hContext TCS上下文句柄。

——KeyUUID 密钥信息的UUID。

输出参数描述：

——pcKeySize　返回密钥blob的字节长度。

——prgbKey　　返回指针，指向包含密钥blob的指针。

返回参数：

TCS_SUCCESS

TCS_E_KEY_NOT_REGISTERED

TCS_E_FAIL

7.1.2.6 Tcs_GetRegisteredKeyByPublicInfo

功能描述：

通过公钥数据在密钥数据库中查找相应的密钥数据。

接口定义：

```
TSM_RESULT Tcs_GetRegisteredKeyByPublicInfo
(
    TCS_CONTEXT_HANDLE          hContext,               // in
    TSM_ALGORITHM_ID            ulAlgId,                // in
    UINT32                      ulPublicInfoLength,     // in
    BYTE*                       rgbPublicInfo,          // in
    UINT32*                     pcKeySize,              //out
    BYTE**                      prgbkey                 //out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。
——ulAlgId　　密钥算法ID。
——ulPublicInfoLength　公钥数据长度。
——rgbPublicInfo　　公钥数据区。

输出参数描述：

——pcKeySize　返回密钥blob的字节长度。
——prgbKey　　返回指针，指向包含密钥blob的指针。

返回参数：

TCS_SUCCESS
TCS_E_KEY_NOT_REGISTERED
TCS_E_FAIL

7.1.2.7 Tcs_CollatePekRequest

功能描述：

创建PEK请求。

接口定义：

```
TSM_RESULT Tcs_CollatePekRequest
(
    TCS_CONTEXT_HANDLE      hContext,                       // in
    TCM_CHOSENID_HASH       IDLabel_PrivCAHash,             // in
    UINT32*                 pcEndorsementCredentialSize,    // out
    BYTE**                  prgbEndorsementCredential,      // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。
——IDLabel_PrivCAHash　PEK信息和CA的摘要。

输出参数描述：

——pcEndorsementCredentialSize　　返回的EK证书长度。
——prgbEndorsementCredential　　返回的内存指针，包含EK证书。

返回参数：

TCS_SUCCESS
TCS_E_FAIL

7.1.3 事件管理

7.1.3.1 Tcs_LogPcrEvent

功能描述：
将PCR事件记录到TCS的PCR事件管理器中。
接口定义：

```
TSM_RESULT Tcs_LogPcrEvent
(
    TCS_CONTEXT_HANDLE          hContext,    // in
    TSM_PCR_EVENT               Event,       // in
    UINT32*                     pNumber      // out
);
```

输入参数描述：
——hContext　TCS上下文句柄。
——Event　事件日志的细节。
输出参数描述：
——pNumber 返回TCS记录该PCR事件的索引号码。TCS为每个PCR事件从0开始编号码，单调递增。
返回参数：
TCS_SUCCESS
TCS_E_BAD_INDEX
TCS_E_BAD_PARAMETER
TCS_E_OUTOFMEMORY
TCS_E_FAIL

7.1.3.2 Tcs_GetPcrEvent

功能描述：
取得TCS中PCR事件管理器中指定PCR的事件信息。如果pNumber有具体值则返回指定的PCR事件信息，如果为NULL，则返回指定PCR的所有事件信息。
接口定义：

```
TSM_RESULT Tcs_GetPcrEvent
(
    TCS_CONTEXT_HANDLE      hContext,    // in
    UINT32                  PcrIndex,    // in
    UINT32*                 pNumber,     // in, out
    TSM_PCR_EVENT**         ppEvent      // out
);
```

输入参数描述：
——hContext TCS上下文句柄。
——PcrIndex PCR索引。

——pNumber 事件编号，指定要返回的PCR事件，具体编码方式由厂商自定义。
输出参数描述：
——pNumber 如果输入时为NULL，则返回的值是所有PCR事件的个数。
——ppEvent PCR事件信息的指针。
返回参数：
TCS_SUCCESS
TCS_E_BAD_INDEX
TCS_E_SIZE
TCS_E_FAIL

7.1.3.3 Tcs_GetPcrEventsByPcr

功能描述：
取得指定PCR的从指定事件开始后的PCR事件信息。
接口定义：

```
TSM_RESULT Tcs_GetPcrEventsByPcr
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    UINT32                      PcrIndex,       // in
    UINT32                      FirstEvent,     // in
    UINT32*                     pEventCount,    // in, out
    TSM_PCR_EVENT**             ppEvents        // out
);
```

输入参数描述：
——hContext　　TCS上下文句柄。
——PcrIndex　　PCR索引。
——FirstEvent　　指定序列中第一个事件的编号。
——pEventCount　　要返回的事件个数。
输出参数描述：
——pEventCount 实际返回的事件个数。
——ppEvents　　返回的PCR事件的TSM_PCR_EVENT结构链表。
返回参数：
TCS_SUCCESS
TCS_E_BAD_INDEX
TCS_E_BAD_PARAMETER
TCS_E_SIZE
TCS_E_FAIL

7.1.3.4 Tcs_GetPcrEventLog

功能描述：
返回TCS保存的所有PCR事件。
接口定义：

```
TSM_RESULT Tcs_GetPcrEventLog
```

```
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    UINT32*                 pEventCount,    // out
    TSM_PCR_EVENT**         ppEvents        // out
);
```

输入参数描述：

——hContext TCS上下文句柄。

输出参数描述：

——pEventCount 实际返回的事件个数。

——ppEvents 指向事件日志数据结构的头指针。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

TCS_E_INVALID_CONTEXTHANDLE

7.2 可信密码模块管理

7.2.1 TCM 测试

7.2.1.1 Tcsip_SelfTestFull

功能描述：

该命令测试TCM的全部功能能否正常运行。

接口定义：

```
TSM_RESULT Tcsip_SelfTestFull
(
    TCS_CONTEXT_HANDLE  hContext    // in
);
```

输入参数描述：

——hContext 创建上下文对象的句柄。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

TCS_E_INVALID_CONTEXTHANDLE

7.2.1.2 Tcsip_ContinueSelfTest

功能描述：

测试TCM初始化时未被测试的模块。

接口定义：

```
TSM_RESULT Tcsip_ContinueSelfTest
(
    TCS_CONTEXT_HANDLE          hContext   // in
);
```

输入参数描述：
——hContext 创建上下文对象的句柄。
输出参数描述：
——无。
返回参数：
TCS_SUCCESS
TCS_E_FAIL

7.2.1.3 Tcsip_GetTestResult

功能描述：
该命令提供自检结果信息。
接口定义：

```
TSM_RESULT Tcsip_GetTestResult
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    UINT32*                     outDataSize,    // out
    BYTE**                      outData         // out
);
```

输入参数描述：
——hContext 创建上下文对象的句柄。
输出参数描述：
——outDataSize 返回的厂商定义的数据长度。
——outData 返回的厂商定义的数据。
返回参数：
TCS_SUCCESS
TCS_E_FAIL
TCS_E_INVALID_CONTEXTHANDLE

7.2.2 工作模式设置

7.2.2.1 Tcsip_SetOwnerInstall

功能描述：
当TCM处于使能状态且没有所有者的情况下，该命令在确认物理现场后，设置TCM允许或拒绝创建所有者。
接口定义：

```
TSM_RESULT Tcsip_SetOwnerInstall
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    TSM_BOOL                    state           // in
);
```

输入参数描述：
——hContext 上下文对象句柄。
——state 状态位，为一个布尔值：TRUE 表示允许 TCM 创建所有者，FALSE 则相反。
返回参数：

TCS_SUCCESS
TCS_E_INVALID_CONTEXTHANDLE

7.2.2.2 Tcsip_OwnerSetDisable

功能描述：
所有者设置TCM处于使能或禁用状态。
接口定义：

```
TSM_RESULT Tcsip_OwnerSetDisable
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_BOOL                disableState,   // in
    TCM_AUTH*               ownerAuth       // in, out
);
```

输入参数描述：
——hContext　　上下文对象句柄。
——disableState　　状态位，标识使能(TRUE)或禁用(FALSE)TCM。
——ownerAuth　　指向授权数据验证码的指针。
输出参数描述：
——ownerAuth　　指向授权数据验证码的指针。
返回参数：
TCS_SUCCESS
TCS_E_INVALID_CONTEXTHANDLE

7.2.2.3 Tcsip_PhysicalEnable

功能描述：
使用物理现场作为授权使能TCM。
接口定义：

```
TSM_RESULT Tcsip_PhysicalEnable
(
    TCS_CONTEXT_HANDLE  hContext        // in
);
```

输入参数描述：
——hContext上下文对象的句柄。
输出参数描述：
——无。
返回参数：
TCS_SUCCESS
TCS_E_INVALID_CONTEXTHANDLE

7.2.2.4 Tcsip_PhysicalDisable

功能描述：
使用物理存在作为授权禁用TCM。
接口定义：

```
TSM_RESULT Tcsip_PhysicalDisable
(
    TCS_CONTEXT_HANDLE  hContext    // in
);
```

输入参数描述：

——hContext 上下文对象的句柄。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_INVALID_CONTEXTHANDLE

7.2.2.5 Tcsip_SetTempDeactivated

功能描述：

TCM的操作者使TCM暂时无效，下一次平台启动时TCM恢复到有效状态。该命令的授权可以是物理现场也可以是操作者授权。

接口定义：

```
TSM_RESULT Tcsip_SetTempDeactived
(
    TCS_CONTEXT_HANDLE  hContext        // in
);
```

输入参数描述：

——hContext 上下文对象的句柄。

返回参数：

TCS_SUCCESS

TCS_E_INVALID_CONTEXTHANDLE

7.2.2.6 Tcsip_PhysicalSetDeactivated

功能描述：

用于设置TCM是否能够使用物理现场作为授权方式。

接口定义：

```
TSM_RESULT Tcspi_PhysicalSetDeactivated
(
    TCS_CONTEXT_HANDLE          hContext,   // in
    TSM_BOOL                    state       // in
);
```

输入参数描述：

——hContext　上下文对象的句柄。

——state　　状态位为是否设置物理现场作为授权标识的状态值,TRUE表明可以使用物理现场作为授权方式，FALSE表明不可以。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.2.2.7 Tcsip_SetOperatorAuth

功能描述：

用于设置TCM的操作者的授权数据。

接口定义：

```
TSM_RESULT Tcspi_SetOperatorAuth
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_SECRET              operatorAuth        // in
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——operatorAuth　　操作者的授权数据。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_INVALID_CONTEXTHANDLE

7.2.2.8 Tcsip_PhysicalPresence

功能描述：

TCM某些命令操作需要物理现场，用来保证平台所有者身份证明或非远程软件对TCM的操作。这个命令有2个功能，第一个是启用或永久使用硬件/软件物理现场；另一个是如果启用软件物理现场后，是否允许使用物理现场。

接口定义：

```
TSM_RESULT Tcspi_PhysicalPresence
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_PHYSICAL_PRESENCE   fPhysicalPresence   // in
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——fPhysicalPresence　　状态为可以设置的物理现场各种状态。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.3　所有者管理

7.2.3.1 Tcsip_TakeOwnership

功能描述：

用于在TCM内部创建所有者的命令，平台所有者只能有一个。

接口定义：

```
TSM_RESULT Tcspi_TakeOwnership
(
    TCS_CONTEXT_HANDLE          hContext,           // in
    UINT16                      protocolID,         // in
    UINT32                      encOwnerAuthSize,   // in
    BYTE*                       encOwnerAuth,       // in
    UINT32                      encSmkAuthSize,     // in
    BYTE*                       encSmkAuth,         // in
    UINT32                      smkKeyInfoSize,     // in
    BYTE*                       smkKeyInfo,         // in
    TSM_AUTH*                   ownerAuth,          // in, out
    UINT32*                     smkKeyDataSize,     // out
    BYTE**                      smkKeyData          // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——protocolID　　协议ID为使用协议类型，这里等于TCM_PID_OWNER。

——encOwnerAuthSize　　加密后的所有者授权数据大小。

——encOwnerAuth　　加密后的所有者授权数据。

——encSmkAuthSize　　加密后的SMK授权数据大小。

——encSmkAuth　　加密后的存储主密钥授权数据。

——smkKeyInfoSize　　TCM_KEY字节流长度。

——smkKeyInfo　　带有SMK创建的密钥参数的TCM_KEY结构。

——ownerAuth　　TCM所有者授权。

输出参数描述：

——ownerAuth　　输出TCM所有者授权。

——smkKeyDataSize　　创建所有者后SMK密钥数据长度。

——smkKeyData　　创建所有者后SMK密钥数据。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.3.2 Tcsip_OwnerClear 和 Tcsip_ForceClear

所有者授权下清除 Tcsip_OwnerClear

功能描述：

该命令在所有者授权下执行清除操作。

接口定义：

```
TSM_RESULT Tcspi_OwnerClear
(
    TCS_CONTEXT_HANDLE      hContext,           // in
```

```
        TSM_AUTH*                ownerAuth,          // in, out
    );
```

输入参数描述：

——hContext　上下文对象的句柄。

——ownerAuth　TCM所有者授权。

输出参数描述：

——ownerAuth 输出TCM所有者授权。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

物理在线授权下清除 Tcsip_ForceClear

功能描述：

该命令在物理现场的条件下执行清除所有者操作。

接口定义：

```
TSM_RESULT Tcspi_ForceClear
(
    TCS_CONTEXT_HANDLE  hContext,           // in
);
```

输入参数描述：

——hContext 上下文对象的句柄。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.3.3 Tcsip_DisableOwnerClear 和 Tcsip_DisableForceClear

禁止所有者授权下的清除 Tcsip_DisableOwnerClear

功能描述：

该命令使TCM_OwnerClear命令无效。

接口定义：

```
TSM_RESULT Tcspi_DisableOwnerClear
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_AUTH*               ownerAuth,          // in, out
);
```

输入参数描述：

——hContext　上下文对象的句柄。

——ownerAuth TCM所有者授权。

输出参数描述：

——ownerAuth 输出 TCM 所有者授权。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

禁止物理在线授权下的清除 Tcsip_DisableForceClear

功能描述：

限制使用Tcspi_ForceClear命令的使用。

接口定义：

```
TSM_RESULT Tcspi_DisableForceClear
(
    TCS_CONTEXT_HANDLE  hContext    // in
);
```

输入参数描述：

——hContext 上下文对象的句柄。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.4 属性管理

7.2.4.1 Tcsip_GetCapability

功能描述：

该命令返回TCM的当前属性信息。

接口定义：

```
TSM_RESULT Tcspi_GetCapability
(
    TCS_CONTEXT_HANDLE          hContext,    // in
    TSM_CAPABILITY_AREA         capArea,     // in
    UINT32                      subCapSize,  // in
    BYTE*                       subCap,      // in
    UINT32*                     respSize,    // out
    BYTE**                      resp         // out
);
```

输入参数描述：

——hContext 上下文对象的句柄。

——capArea 属性域参数，参考TCM_CAPABILITY_AREA定义。

——subCapSize 子属性参数长度。

——subCap 子属性参数，参考TCM_CAPABILITY_AREA定义。

输出参数描述：

——respSize 返回属性信息值的长度。

——resp 返回属性信息值。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.4.2 Tcsip_SetCapability

功能描述：

命令用于设置TCM的属性值。

接口定义：

```
TSM_RESULT Tcsip_SetCapability
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCM_CAPABILITY_AREA     capArea,        // in
    UINT32                  subCapSize,     // in
    BYTE*                   subCap,         // in
    UINT32                  valueSize,      // in
    BYTE*                   value,          // in
    TCM_AUTH*               ownerAuth       // in out
);
```

输入参数描述：

——hContext　上下文对象句柄。

——capArea　属性域参数。

——subCapSize　子属性参数长度。

——subCap　子属性参数。

——valueSize　属性值长度。

——value　属性值。

——ownerAuth　所有者授权数据，可以为NULL。

输出参数描述：

——ownerAuth 如果操作成功，为返回TCM计算的授权数据。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.5 升级与维护

7.2.5.1 Tcsip_FieldUpgrade

功能描述：

用于升级TCM的固件。各厂商可自行定义相关数据结构解释输入与输出数据。

接口定义：

```
TSM_RESULT Tcsip_FieldUpgrade
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    UINT32                      dataInSize,     // in
    BYTE*                       dataIn,         // in
    TCM_AUTH*                   ownerAuth,      // in, out
    UINT32*                     dataOutSize,    // out
    BYTE**                      dataOut         // out
);
```

输入参数描述：

——hContext　上下文对象句柄。

——dataInSize　升级输入数据的长度。

——dataIn　升级输入数据。

——ownerAuth　所有者授权信息指针。

输出参数描述：

——ownerAuth　如果操作成功，为返回TCM计算的授权信息。

——dataOutSize 升级输出数据的长度。

——dataOut　　升级输出数据。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.5.2 Tcsip_ResetLockValue

功能描述：

该命令用于重置TCM字典攻击次数。

接口定义：

```
TSM_RESULT Tcsip_ResetLockValue
(
    TCS_CONTEXT_HANDLE      hContext,    // in
    TCM_AUTH*               ownerAuth    // in, out
);
```

输入参数描述：

——hContext　　上下文对象句柄。

——ownerAuth　　所有者授权信息指针。

输出参数描述：

——ownerAuth 如果操作成功，为返回TCM计算的授权信息。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.6　授权管理

7.2.6.1 Tcsip_ChangeAuth

功能描述：

本命令允许一个实体的所有者改变这个实体的授权数据。

接口定义：

```
TSM_RESULT Tcsip_ChangeAuth
(
    TCS_CONTEXT_HANDLE      hContext,        // in
    TCS_KEY_HANDLE          parentHandle,    // in
    TCM_PROTOCOL_ID         protocolID,      // in
    TCM_ENCAUTH             newAuth,         // in
    TCM_ENTITY_TYPE         entityType,      // in
    UINT32                  encDataSize,     // in
    BYTE*                   encData,         // in
    TCM_AUTH*               ownerAuth,       // in, out
    TCM_AUTH*               entityAuth,      // in, out
    UINT32*                 outDataSize,     // out
```

```
    BYTE**                  outData         // out
);
```

输入参数描述：

——hContext　上下文对象句柄。

——parentHandle 实体的父密钥句柄。

——protocolID　使用的协议 ID。

——newAuth　加密后的新实体授权数据。

——entityType　需要改变实体的实体类型。

——encDataSize 加密后实体数据的长度。

——encData　需要改变实体的被加密后的实体数据,实体类型为 TCM_ET_DATA,TCM_ET_KEY 中一种。

——ownerAuth　所有者授权信息指针。

——entityAuth　实体授权信息指针。

输出参数描述：

——ownerAuth　返回所有者授权的授权信息指针。

——entityAuth　返回实体授权的授权信息指针。

——outDataSize 加密的实体数据的长度。

——outData　为改变后的，经过加密的实体数据。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.6.2 Tcsip_ChangeAuthOwner

功能描述：

本命令TCM所有者改变所有者或SMK的授权数据。

接口定义：

```
TSM_RESULT Tcsip_ChangeAuthOwner
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    TCM_PROTOCOL_ID             protocolID,     // in
    TCM_ENCAUTH                 newAuth,        // in
    TCM_ENTITY_TYPE             entityType,     // in
    TCM_AUTH*                   ownerAuth       // in, out
);
```

输入参数描述：

——hContext　上下文对象句柄。

——protocolID　使用的协议 ID。

——newAuth　加密传输授权实体授权数据。

——entityType　需要改变实体的实体类型。

——ownerAuth　所有者授权信息指针。

输出参数描述：

——ownerAuth 为返回TCM计算的授权信息。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.7 非易失性存储管理

7.2.7.1 Tcsip_NV_DefineOrReleaseSpace

功能描述：

本函数创建或释放NV空间。

在TCM内定义或者释放一个NV区，定义时同时设置对该NV的读写方法和读取大小。如果对一个已经存在的NV区再次调用该命令，并且NV区大小为0，TCM将删除该NV区。

如果对一个已经存在的NV区再次调用该命令，并且NV区大小不为0，对应的nv大小按照新的定义的空间大小重新定义。

接口定义：

```
TSM_RESULT Tcsip_NV_DefineOrReleaseSpace
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCM_UINT32              cPubInfoSize,   // in
    BYTE*                   pPubInfo,       // in
    TCM_ENCAUTH             encAuth,        // in
    TCM_AUTH*               pAuth           // in, out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——cPubInfoSize　　公共参数大小。

——pPubInfo　　请求的NV区的公共参数。

——encAuth　　使用该NV区时的授权数据。

——pAuth　　TCM拥有者授权。如果为NULL，则不需所有者授权。

输出参数描述：

——pAuth TCM拥有者授权。如果为NULL，则不需所有者授权。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.2.7.2 Tcsip_NV_WriteValue 和 Tcsip_NV_WriteValueAuth

非授权写入 Tcsip_NV_WriteValue

功能描述：

本函数往NV空间写入数据。

接口定义：

```
TSM_RESULT Tcsip_NV_WriteValue
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_NV_INDEX            hNVStore,       // in
```

```
    UINT32                  offset,          // in
    UINT32                  ulDataLength,    // in
    BYTE*                   rgbDataToWrite,  // in
    TCM_AUTH*               privAuth         // in, out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——hNVStore　　TCM内的NV区索引。

——offset　　NV空间的偏移量。

——ulDataLength　　要写入的数据长度。

——rgbDataToWrite　　写入的数据。

——privAuth　　所有者授权。如果为NULL，则不需所有者授权。

输出参数描述：

——privAuth　　所有者授权。如果为NULL，则不需所有者授权。

返回参数：

TSM_SUCCESS
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_E_BAD_INDEX
TSM_MAXNVWRITE
TSM_AUTH_CONFLICT
TSM_AUTHFAIL
TSM_AREA_LOCKED
TSM_BAD_LOCALITY
TSM_BAD_PRESENCE
TSM_DISABLED_CMD
TSM_NOSPACE
TSM_NOT_FULLWRITE
TSM_WRONGPCRVALUE

授权写入 Tcsip_NV_WriteValueAuth

功能描述：

本函数往NV空间写入数据，需要验证NV空间的授权数据。

接口定义：

```
TSM_RESULT Tcsip_NV_WriteValueAuth
(
    TCS_CONTEXT_HANDLE      hContext,        // in
    TSM_NV_INDEX            hNVStore,        // in
    UINT32                  offset,          // in
    UINT32                  ulDataLength,    // in
    BYTE*                   rgbDataToWrite,  // in
    TCM_AUTH*               NVAuth           // in, out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——hNVStore　　TCM内的NV区索引。

——offset　　NV空间的偏移量。

——ulDataLength　　要写入的数据长度。

——rgbDataToWrite　写入的数据。

——NVAuth　　NV空间授权会话验证信息。此命令无需所有者授权。

输出参数描述：

——NVAuth NV空间授权会话验证信息。此命令无需所有者授权。

返回参数：

TSM_SUCCESS
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_BAD_INDEX
TSM_MAXNVWRITE
TSM_AUTH_CONFLICT
TSM_AUTHFAIL
TSM_AREA_LOCKED
TSM_BAD_LOCALITY
TSM_BAD_PRESENCE
TSM_DISABLED_CMD
TSM_NOSPACE
TSM_NOT_FULLWRITE
TSM_WRONGPCRVALUE

7.2.7.3 Tcsip_NV_ReadValue 和 Tcsip_NV_ReadValueAuth

授权读 Tcsip_NV_ReadValue

功能描述：

本函数使用Owner授权从NV空间读取数据。

接口定义：

```
TSM_RESULT Tcsip_NV_ReadValue
(
      TCS_CONTEXT_HANDLE          hContext,       // in
      TSM_NV_INDEX                hNVStore,       // in
      UINT32                      offset,         // in
      UINT32*                     pulDataLength,  // in, out
      TCM_AUTH*                   privAuth,       // in, out
      BYTE**                      rgbDataRead     // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——hNVStore　　TCM内的NV区索引。

——offset　　NV空间的偏移量。

——pulDataLength　要读取的数据长度。
——privAuth　所有者授权。如果为NULL，则不需所有者授权。
输出参数描述：
——pulDataLength　读取到的数据长度。
——privAuth　所有者授权。如果为NULL，则不需所有者授权。
——rgbDataRead　返回的数据。
返回参数：
TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_BAD_INDEX
TSM_AUTH_CONFLICT
TSM_AUTHFAIL
TSM_BAD_LOCALITY
TSM_BAD_PRESENCE
TSM_DISABLED_CMD
TSM_NOSPACE
TSM_WRONGPCRVALUE
授权读取 Tcsip_NV_ReadValueAuth
功能描述：
本函数使用NV空间授权从NV空间读取数据。
接口定义：

```
TSM_RESULT Tcsip_NV_ReadValueAuth
 (
    TCS_CONTEXT_HANDLE     hContext,        // in
    TSM_NV_INDEX           hNVStore,        // in
    UINT32                 offset,          // in
    UINT32*                pulDataLength,   // in, out
    TCM_AUTH*              NVAuth,          // in, out
    BYTE**                 rgbDataRead      // out
 );
```

输入参数描述：
——hContext　TCS上下文句柄。
——hNVStore　TCM内的NV区索引。
——offset　NV空间的偏移量。
——pulDataLength　要读取的数据长度。
——NVAuth　NV空间授权会话验证信息。此命令无需所有者授权。
输出参数描述：
——pulDataLength 读取到的数据长度。
——NVAuth NV空间授权会话验证信息。此命令无需所有者授权。
——rgbDataRead 返回的数据。
返回参数：

TSM_SUCCESS
TSM_E_INVALID_HANDLE
TSM_E_BAD_PARAMETER
TSM_E_INTERNAL_ERROR
TSM_BAD_INDEX
TSM_AUTH_CONFLICT
TSM_AUTHFAIL
TSM_BAD_LOCALITY
TSM_BAD_PRESENCE
TSM_DISABLED_CMD
TSM_NOSPACE
TSM_WRONGPCRVALUE

7.2.8 审计

7.2.8.1 Tcsip_GetAuditDigest

功能描述：

该命令返回当前的审计摘要以及审计列表。

接口定义：

```
TSM_RESULT Tcsip_GetAuditDigest
(
    TCS_CONTEXT_HANDLE          hContext,               // in
    UINT32                      startOrdinal,           // in
    TCM_DIGEST*                 auditDigest,            // out
    UINT32*                     counterValueSize,       // out
    BYTE**                      counterValue,           // out
    TSM_BOOL*                   more,                   // out
    UINT32*                     ordSize,                // out
    UINT32**                    ordList                 // out
);
```

输入参数描述：

——hContext　　创建上下文对象的句柄。
——startOrdinal　　开始序列号，表明从何处开始返回审计列表.

输出参数描述：

——auditDigest　　审计事件的摘要。
——counterValueSiz　　参数 counterValue 缓冲大小。
——counterValue　　返回设计单调计数器当前值。
——more　　是否全部返回的标记。
——ordSize　　审计顺序列表中序数大小。
——ordList　　指明返回命令列表是否包含所有请求命令。

返回参数：

TCS_SUCCESS
TCS_E_NOTIMPL

7.2.8.2 Tcsip_GetAuditDigestSigned

功能描述：

该命令返回当前的审计命令列表及其签名。

接口定义：

```
TSM_RESULT Tcsip_GetAuditDigestSigned
(
    TCS_CONTEXT_HANDLE          hContext,           // in
    TCS_KEY_HANDLE              keyHandle,          // in
    TSM_BOOL                    closeAudit,         // in
    TCM_NONCE                   antiReplay,         // in
    TCM_AUTH*                   privAuth,           // in, out
    UINT32*                     counterValueSize,   // out
    BYTE**                      counterValue,       // out
    TCM_DIGEST*                 auditDigest,        // out
    TCM_DIGEST*                 ordinalDigest,      // out
    UINT32*                     sigSize,            // out
    BYTE**                      sig                 // out
);
```

输入参数描述：

——hContext　创建上下文对象的句柄。

——keyHandle　执行数字签名的加载密钥的密钥句柄标志。

——closeAudit　用于标记签名摘要信息是否重新开始计算。

——antiReplay　数字签名操作中的抗重放 nouce。

——privAuth　授权使用密钥句柄的授权摘要。

输出参数描述：

——privAuth　授权使用密钥句柄的授权摘要。

——counterValueSize　参数 counterValue 缓冲大小。

——counterValue　审计计数器的值。

——auditDigest　TCM 审计摘要。

——ordinalDigest　审计顺序列表摘要。

——sigSize　返回的数字签名长度。

——sig　最终的数字签名。

返回值：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.8.3 Tcsip_SetOrdinalAuditStatus

功能描述：

设置一个给定命令的审计标志，必须判断指定命令是否可被设置。

接口定义：

```
TSM_RESULT Tcsip_SetOrdinalAuditStatus
(
```

```
        TCS_CONTEXT_HANDLE      hContext,        // in
        TCM_AUTH*               ownerAuth,       // in, out
        UINT32                  ordinalToAudit,  // in
        TSM_BOOL                auditState       // in
    );
```

输入参数描述：

——hContext　　上下文对象句柄。

——ownerAuth　　所有者授权信息指针，不能为 NULL。

——ordinalToAudit　将要设置的命令码。

——auditState　　“1”代表需要被审计，“0”代表不需要被审计。

输出参数描述：

——ownerAuth　　为返回所有者授权数据，不能为NULL。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.9　时钟

7.2.9.1 Tcsip_ReadCurrentTicks

功能描述：

获取TCM的当前时钟节拍。

接口定义：

```
TSM_RESULT Tcsip_ReadCurrentTicks
(
      TCS_CONTEXT_HANDLE    hContext,           // in
      UINT32*               pulCurrentTime,     // out
      BYTE**                prgbCurrentTime     // out
);
```

输入参数描述：

——hContext 上下文对象句柄。

输出参数描述：

——pulCurrentTime　返回prgbCurrentTime的长度。

——prgbCurrentTime 返回TCM中的当前时间计数数据(TCM_CURRENT_TICKSstruct)。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.9.2 Tcsip_TickStampBlob

功能描述：

对一块数据进行时间戳操作。

接口定义：

```
TSM_RESULT Tcsip_TickStampBlob
(
    TCS_CONTEXT_HANDLE      hContext,           // in
```

```
        TSM_HKEY                hKey,                // in
        TCM_NONCE               antiReplay,          // in
        TCM_DIGEST              digestToStamp,       // in
        TCM_AUTH*               privAuth,            // in, out
        UINT32*                 pulSignatureLength,  // out
        BYTE**                  prgbSignature,       // out
        UINT32*                 pulTickCountLength,  // out
        BYTE**                  prgbTickCount        // out
    );
```

输入参数描述：

——hContext　　上下文对象句柄。

——hKey　　签名密钥句柄。

——antiReplay　　反重放攻击数据。

——digestToStamp　　需要被执行之间戳的数据。

——privAuth　　签名密钥授权信息指针，可以为 NULL。

输出参数描述：

——privAuth　　为返回签名密钥授权信息，可以为 NULL。

——pulSignatureLength　　为返回 prgbSignature 的长度。

——prgbSignature　　执行时间戳后的签名。

——pulTickCountLength　　为返回 prgbTickCount 的长度。

——prgbTickCount　　为返回TCM中的当前时间计数数据(TCM_CURRENT_TICKSstruct)。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.10　计数器

7.2.10.1　Tcsip_CreateCounter

功能描述：

创建一个新的单调计数器，并赋予这个计数器授权数据与标签。

接口定义：

```
    TSM_RESULT Tcsip_CreateCounter
    (
        TCS_CONTEXT_HANDLE      hContext,        // in
        UINT32                  LabelSize,       // in
        BYTE *                  pLabel,          // in
        TCM_ENCAUTH             CounterAuth,     // in
        TCM_AUTH *              pOwnerAuth,      // in, out
        TSM_COUNTER_ID *        idCounter,       // out
        TCM_COUNTER_VALUE *     pCounterValue    // out
    );
```

输入参数描述：

——hContext　　上下文对象句柄。

——LabelSize　　标签长度。

——pLabel　　计数器的标签。
——CounterAuth　　被加密的计数器授权数据。
——pOwnerAuth　　所有者授权信息指针，不能为 NULL。
输出参数描述：
——pOwnerAuth　　返回 TCM 计算的授权信息，不能为 NULL。
——idCounter　　新创建的计数器的 ID。
——pCounterValue　　计数器的初始值。
返回参数：
TCS_SUCCESS
TCS_E_NOTIMPL

7.2.10.2 Tcsip_IncrementCounter

功能描述：
将一个计数器的值增加1，并且选择这个计数器。
接口定义：

```
TSM_RESULT Tcsip_IncrementCounter
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_COUNTER_ID          idCounter,      // in
    TCM_AUTH *              pCounterAuth,   // in, out
    TCM_COUNTER_VALUE *     pCounterValue   // out
);
```

输入参数描述：
——hContext　　上下文对象句柄。
——idCounter　　需要增加/选定的计数器句柄。
——pCounterAuth　　计数器授权数据，不能为 NULL。
输出参数描述：
——pCounterAuth　　为返回计数器授权数据，不能为 NULL。
——pCounterValue　　增加后的计数器的值。
返回参数：
TCS_SUCCESS
TCS_E_NOTIMPL

7.2.10.3 Tcsip_ReadCounter

功能描述：
读取指定计数器的计数值。
接口定义：

```
TSM_RESULT Tcsip_ReadCounter
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TSM_COUNTER_ID          idCounter,          // in
    TCM_COUNTER_VALUE*      counterValue        // out
);
```

输入参数描述：

——hContext　TCS上下文句柄。

——idCounter　待读取的计数器编号。

输出参数描述：

——counterValue 计数器的当前值。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

7.2.10.4 Tcsip_ReleaseCounter 和 Tcsip_ReleaseCounterOwner

非授权释放计数器 Tcsip_ReleaseCounter

功能描述：

非授权释放计数器。

接口定义：

```
TSM_RESULT Tcsip_ReleaseCounter
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_COUNTER_ID          idCounter,      // in
    TCM_AUTH *              pCounterAuth    // in, out
);
```

输入参数描述：

——hContext　TCS上下文句柄。

——idCounter　释放的计数器ID。

——pCounterAuth　计数器授权数据。

输出参数描述：

——pCounterAuth 计数器授权数据。

返回参数：

TCS_SUCCESS

TCS_E_NOTIMPL

使用 Owner 授权释放计数器 Tcsip_ReleaseCounterOwner

功能描述：

使用Owner授权数据释放计数器。

接口定义：

```
TSM_RESULT Tcsip_ReleaseCounterOwner
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TSM_COUNTER_ID          idCounter,      // in
    TCM_AUTH *              pOwnerAuth      // in, out
);
```

输入参数描述：

——hContext　TCS上下文句柄。

——idCounter　　要释放的计数器ID。
——pOwnerAuth　　所有者授权数据。
输出参数描述：
——pOwnerAuth　　所有者授权数据。
返回参数：
TCS_SUCCESS
TCS_E_FAIL
TSM_E_INTERNAL_ERROR

7.3 平台身份标识与认证

7.3.1 密码模块密钥管理

7.3.1.1 Tcsip_CreateEndorsementKeyPair

功能描述：
创建不可撤消的密码模块密钥EK。
接口定义：

```
TSM_RESULT Tcsip_CreateEndorsementKeyPair
(
    TCS_CONTEXT_HANDLE      hContext,                   // in
    TCM_NONCE               antiReplay,                 // in
    UINT32                  endorsementKeyInfoSize,     // in
    BYTE*                   endorsementKeyInfo,         // in
    UINT32*                 endorsementKeySize,         // out
    BYTE**                  endorsementKey,             // out
    TCM_DIGEST*             checksum                    // out
);
```

输入参数描述：
——hContext　　TCS上下文句柄
——antiReplay　　防重放攻击Nonce
——endorsementKeyInfoSize　　产生EK密钥的参数长度
——endorsementKeyInfo　　产生EK密钥的参数
输出参数描述：
——endorsementKeySize　　EK公钥长度
——EndorsementKey　　EK公钥
——Checksum　　校验值
返回参数：
TCS_SUCCESS
TCS_E_FAIL

7.3.1.2 Tcsip_CreateRevocableEndorsementKeyPair

功能描述：
创建可撤消的密码模块密钥EK。

接口定义：

```
TSM_RESULT Tcsip_CreateRevocableEndorsementKeyPair
(
    TCS_CONTEXT_HANDLE  hContext,                // in
    TCM_NONCE           antiReplay,              // in
    UINT32              endorsementKeyInfoSize,  // in
    BYTE*               endorsementKeyInfo,      // in
    TSM_BOOL            GenResetAuth,            // in
    TCM_DIGEST*         EKResetAuth,             // in, out
    UINT32*             endorsementKeySize,      // out
    BYTE**              endorsementKey,          // out
    TCM_DIGEST*         checksum                 // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄

——antiReplay　　防重放攻击Nonce

——endorsementKeyInfoSize　　产生ＥＫ密钥的参数长度

——endorsementKeyInfo　　产生ＥＫ密钥的参数

——GenResetAuth　　如果为TRUE，则随机产生，否则使用输入值的EKResetAuth

——EKResetAuth　　用于撤销EK时验证

输出参数描述：

——EKResetAuth　　用于撤销EK时验证

——endorsementKeySize　　EK公钥长度

——EndorsementKey　　EK公钥

——Checksum　　校验值

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.3.1.3 Tcsip_RevokeEndorsementKeyPair

功能描述：

撤销密码模块密钥EK。

接口定义：

```
TSM_RESULT Tcsip_RevokeEndorsementKeyPair
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCM_DIGEST              EKResetAuth     // in
);
```

输入参数描述：

——hContext　　TCS上下文句柄

——EKResetAuth　　撤销EK的授权值

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.3.1.4 Tcsip_ReadPubEK 和 Tcsip_OwnerReadInternalPub

非授权读取公钥 Tcsip_ReadPubEK

功能描述：

非授权读取密码模块密钥EK的公钥。

接口定义：

```
TSM_RESULT Tcsip_ReadPubek
(
    TCS_CONTEXT_HANDLE      hContext,               // in
    TCM_NONCE               antiReplay,             // in
    UINT32*                 pubEndorsementKeySize,  // out
    BYTE**                  pubEndorsementKey,      // out
    TCM_DIGEST*             checksum                // out
);
```

输入参数描述：

——hContext　　TCS 上下文句柄

——antiReplay　　防重放攻击 Nonce

输出参数描述：

——pubEndorsementKeySize　　EK 公钥长度

——pubEndorsementKey　　EK 公钥

——Checksum　　校验值

返回参数：

TCS_SUCCESS

TCS_E_FAIL

授权读取公钥 Tcsip_OwnerReadInternalPub

功能描述：

所有者授权读取密码模块密钥EK的公钥。

接口定义：

```
TSM_RESULT Tcsip_OwnerReadInternalPub
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCS_KEY_HANDLE          hKey,           // in
    TCM_AUTH*               pOwnerAuth,     // in, out
    UINT32*                 punPubKeySize,  // out
    BYTE**                  ppbPubKeyData   // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——hKey　　EK密钥句柄。

——pOwnerAuth　　Owner授权会话验证信息。

输出参数描述：

——pOwnerAuth　　Owner授权会话验证信息。

——punPubKeySize　EK公钥长度。

——ppbPubKeyData　EK公钥。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.3.2　平台身份密钥管理

7.3.2.1 Tcsip_MakeIdentity

功能描述：

创建PIK密钥及PIK证书请求信息。

接口定义：

```
TSM_RESULT Tcsip_MakeIdentity
(
      TCS_CONTEXT_HANDLE      hContext,                  // in
      TCM_ENCAUTH             identityAuth,              // in
      TCM_CHOSENID_HASH       IDLabel_PrivCAHash,        // in
      UINT32                  idIdentityKeyInfoSize,     // in
      BYTE*                   idIdentityKeyInfo,         // in
      TCM_AUTH*               pSmkAuth,                  // in, out
      TCM_AUTH*               pOwnerAuth,                // in, out
      UINT32*                 idIdentityKeySize,         // out
      BYTE**                  idIdentityKey,             // out
      UINT32*                 pcIdentityBindingSize,     // out
      BYTE**                  prgbIdentityBinding,       // out
      UINT32*                 pcEndorsementCredentialSize, // out
      BYTE**                  prgbEndorsementCredential,  // out
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——identityAuth　　加密的PIK授权数据。

——IDLabel_PrivCAHash　　平台身份标识、可信方公钥(TCM_PUBKEY结构数据)的摘要

——IdIdentityKeyInfoSize　　产生PIK的参数长度。

——idIdentityKeyInfo　　产生PIK的参数，为TCM_KEY结构数据。

——pSmkAuth　　SMK授权会话验证信息。

——pOwnerAuth　　Owner授权会话验证信息。

输出参数描述：

——pSmkAuth　　SMK授权会话验证信息

——pOwnerAuth　　Owner授权会话验证信息

——idIdentityKeySize　　产生的PIK长度

——idIdentityKey　　产生的PIK，为TCM_KEY结构数据
——pcIdentityBindingSize　　prgbIdentityBinding长度
——prgbIdentityBinding　　用PIK私钥对TCM_IDENTITY_CONTENTS结构数据的签名结果
——pcEndorsementCredentialSize　EK证书数据长度。如果等于0，说明EK证书数据为空
——prgbEndorsementCredential　EK证书数据。可以为空，可采用其他方式获取EK证书

返回参数：

TSM_SUCCESS
TSM_E_FAIL
TSM_E_BAD_PARAMETER
TSM_E_INVALID_HANDLE
TSM_E_INVALID_AUTH_SESSION

7.3.2.2 Tcsip_ActivateIdentity

功能描述：

得到加密PIK证书的对称密钥。

接口定义：

```
TSM_RESULT Tcsip_ActivateIdentity
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TCS_KEY_HANDLE          idKey,              // in
    UINT32                  blobSize,           // in
    BYTE*                   blob,               // in
    TCM_AUTH*               idKeyAuth,          // in, out
    TCM_AUTH*               ownerAuth,          // in, out
    UINT32*                 SymmetricKeySize,   // out
    BYTE**                  SymmetricKey        // out
);
```

输入参数描述：

——hContext　TCS上下文句柄
——idKey　PIK 句柄
——BlobSize　blob 数据的大小(单位：字节)
——Blob　可信方用 EK 公钥对 TCM_ASYM_CA_CONTENTS 结构数据的加密结果
——idKeyAuth　PIK 授权会话验证信息
——OwnerAuth　Owner 授权会话验证信息

输出参数描述：

——idKeyAuth　PIK 授权会话验证信息
——OwnerAuth　Owner 授权会话验证信息
——SymmetricKeySize　对称密钥长度
——SymmetricKey　对称密钥

返回参数：

TSM_SUCCESS
TSM_E_FAIL
TSM_E_INVALID_HANDLE

TSM_E_INVALID_AUTH_SESSION

7.3.2.3 Tcsip_ActivatePEKCert

功能描述：

得到加密PEK证书的对称密钥。

接口定义：

```
TSM_RESULT Tcsip_ActivatePEKCert
(
      TCS_CONTEXT_HANDLE      hContext,           // in
      UINT32                  blobSize,           // in
      BYTE*                   blob,               // in
      TCM_AUTH*               ownerAuth,          // in, out
      UINT32*                 SymmetricKeySize,   // out
      BYTE**                  SymmetricKey        // out
);
```

输入参数描述：

——hContext　　TCS 上下文句柄。

——BlobSize　　Blob 的大小(单位：字节)。

——Blob　　可信方用 EK 公钥加密的 TCM_SYMMETRIC_KEY 结构数据。

——OwnerAuth　　Owner 授权会话验证信息。

输出参数描述：

——OwnerAuth　　Owner 授权会话验证信息。

——SymmetricKeySize　　对称密钥长度。

——SymmetricKey　　对称密钥。

返回参数：

TSM_SUCCESS

TSM_E_FAIL

TSM_E_INVALID_HANDLE

TSM_E_INVALID_AUTH_SESSION

7.3.2.4 Tcsip_ActivatePEK

功能描述：

本函数将PEK密钥导入TCM内部。

接口定义：

```
TSM_RESULT Tcsip_ActivatePEK
(
      TCS_CONTEXT_HANDLE      hContext,           // in
      TCM_ENCAUTH             KeyUsageAuth,       // in
      TCM_ENCAUTH             KeyMigrationAuth,   // in
      UINT32                  PEKKeyInfoSize,     // in
      BYTE*                   PEKKeyInfo,         // in
```

```
        UINT32                  PEKData Size,       // in
        BYTE*                   PEKData,            // in
        UINT32*                 EncSymKeySize,      // in
        BYTE**                  EncSymKey,          // in
        TCM_AUTH*               pSmkAuth,           // in, out
        TCM_AUTH*               pOwnerAuth,         // in, out
        UINT32*                 PekKeySize,         // out
        BYTE**                  PekKey,             // out
    );
```

输入参数描述：

——hContext　　上下文对象的句柄。

——KeyUsageAuth　　加密的使用授权数据。

——KeyMigrationAuth 加密的迁移授权数据。

——PEKKeyInfoSize　PEKKeyInfo 长度(单位：字节)。

——PEKKeyInfo　　PEK 参数，为 TCM_KEY 结构数据，用于本地设置 PEK 密钥属性。

——PEKDataSize　　PEKData 的长度(单位：字节)。

——PEKData　　可信方产生的用对称密钥加密的 PEK(TCM_KEY 结构数据)。

——EncSymKeySize　CA 产生的用 EK 加密的对称密钥长度。

——EncSymKey　　CA 产生的用 EK 加密的对称密钥。

——pSmkAuth　　SMK 授权会话验证信息。

——pOwnerAuth　　Owner 授权会话验证信息。

输出参数描述：

——pSmkAuth　　SMK 授权会话验证信息。

——pOwnerAuth　　Owner 授权会话验证信息。

——PekKeySize　　PekKey 长度。

——PekKey　　导入的 PEK，为 TCM_KEY 结构数据，属性部分从 PEKKeyInfo 参数获取，公钥和私钥部分从解密的 PEKData 数据里获取，私钥部分被 SMK 加密。

返回参数：

TSM_SUCCESS

TSM_E_FAIL

TSM_E_INVALID_HANDLE

TSM_E_INVALID_AUTH_SESSION

7.4　平台数据保护

7.4.1　数据保护操作

7.4.1.1 Tcsip_Seal

功能描述：

将数据与特定的平台配置信息（PCR值）及平台验证信息（TCM_Proof）绑定在一起生成封装数据。

接口定义：

```
TSM_RESULT Tcsip_Seal
(
    TCS_CONTEXT_HANDLE hContext,        // in
```

```
    TCS_KEY_HANDLE keyHandle,           // in
    TCM_ENCAUTH encAuth,                // in
    UINT32 pcrInfoSize,                 // in
    BYTE* PcrInfo,                      // in
    UINT32 inDataSize,                  // in
    BYTE* inData,                       // in
    TCM_AUTH* pAuth,                    // in, out
    UINT32* SealedDataSize,             // out
    BYTE** SealedData                   // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄.

——keyHandle　　封装操作密钥的密钥句柄。

——encAuth　　加密的授权数据为被加密的封装对象的授权数据,其中加密密钥为授权会话句柄指向的共享会话密钥。

——pcrInfoSize　　PCR信息参数的长度。如果为0，则表明无可用的PCR寄存器。

——PcrInfo　　PCR信息。

——inDataSize　　待封装数据长度。

——inData　　待封装数据。

——pAuth　　执行封装操作密钥的授权数据验证码。

输出参数描述：

——pAuth　　输出的授权数据验证码。

——SealedDataSize　　被封装数据块的长度。

——SealedData　　被封装数据块。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.1.2 Tcsip_Unseal

功能描述：

当封装数据中的平台配置信息（PCR值）及平台验证信息（TCM_Proof）与当前PCR的值和TCM_Proof的值一致时，将TCM_Seal命令生成的封装数据解密。

接口定义：

```
TSM_RESULT Tcsip_Unseal
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCS_KEY_HANDLE          keyHandle,      // in
    UINT32                  SealedDataSize, // in
    BYTE*                   SealedData,     // in
    TCM_AUTH*               keyAuth,        // in, out
    TCM_AUTH*               dataAuth,       // in, out
    UINT32*                 DataSize,       // out
    BYTE**                  Data            // out
```

```
);
```

输入参数描述：

——hContext　　上下文对象的句柄。
——keyHandle　　执行解封操作的密钥的句柄。
——SealedDataSize　封装数据块的长度。
——SealedData　　由TCM_Seal命令生成的封装数据块。
——keyAuth　　输入的基于解封操作密钥的授权数据验证码。
——dataAuth　　输入的基于封装数据的授权数据验证码。

输出参数描述：

——keyAuth　　输出的基于解封操作密钥的授权数据验证码。
——dataAuth　　输出的基于封装数据的授权数据验证码。
——DataSize　　解封后的被封装数据的长度。
——Data　　解封后的被封装数据。

返回参数：

TCS_SUCCESS
TCS_E_FAIL

7.4.2　密钥管理

7.4.2.1 Tcsip_CreateWrapKey

功能描述：

请求TCM依据输入的TCM_KEY结构要求的密钥属性生成密钥。

接口定义：

```
TSM_RESULT Tcsip_CreateWrapKey
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TCS_KEY_HANDLE          hWrappingKey,       // in
    TCM_ENCAUTH             KeyUsageAuth,       // in
    TCM_ENCAUTH             KeyMigrationAuth,   // in
    UINT32                  keyInfoSize,        // in
    BYTE*                   keyInfo,            // in
    TCM_AUTH*               pAuth,              // in, out
    UINT32*                 keyDataSize,        // out
    BYTE**                  keyData,            // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。
——hWrappingKey　保护操作密钥句柄。
——KeyUsageAuth　加密的使用授权数据。
——KeyMigrationAuth 加密的迁移授权数据。
——keyInfoSize　被创建密钥的信息的长度。
——keyInfo　　被创建密钥的信息。
——pAuth　　保护操作密钥授权数据验证码。

输出参数描述：

——pAuth　　保护操作密钥授权数据验证码。
——keyDataSize 创建的密钥信息的长度。
——keyData　　创建的密钥信息。
返回参数：
　　TCS_SUCCESS
　　TCS_E_FAIL

7.4.2.2 Tcsip_LoadKeyByBlob 和 Tcsip_LoadKeyByUUID

Tcsip_LoadKeyByBlob
功能描述：
　　把加密了的密钥数据块加载到TCM中，返回TCM创建的密钥句柄。
接口定义：

```
TSM_RESULT Tcsip_LoadKeyByBlob
(
    TCS_CONTEXT_HANDLE  hContext,                   // in
    TCS_KEY_HANDLE      hUnwrappingKey,             // in
    UINT32              cWrappedKeyBlobSize,        // in
    BYTE*               rgbWrappedKeyBlob,          // in
    TCM_AUTH*           pAuth,                      // in, out
    TCS_KEY_HANDLE*     phKeyTCSI                   // out
);
```

输入参数描述：
——hContext　　上下文对象的句柄。
——hUnwrappingKey　　保护操作密钥句柄。
——cWrappedKeyBlobSize　　密钥数据块的长度。
——rgbWrappedKeyBlob　　密钥数据块。
——pAuth　　保护操作密钥的授权数据验证码。
输出参数描述：
——pAuth　　保护操作密钥的授权数据验证码。
——phKeyTCSI　　被加载密钥的句柄。
返回参数：
　　TCS_SUCCESS
　　TCS_E_FAIL
Tcsip_LoadKeyByUUID
功能描述：
　　根据密钥管理器的密钥UUID将密钥数据块加载到TCM中，返回TCM创建的密钥句柄。
接口定义：

```
TSM_RESULT Tcsip_LoadKeyByUUID
(
  TCS_CONTEXT_HANDLE        hContext,       // in
  TSM_UUID                  KeyUUID,        // in
  TCS_LOADKEY_INFO*         pLoadKeyInfo,   // in, out
  TCS_KEY_HANDLE*           phKeyTCSI       // out
```

```
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——KeyUUID　　被加载的密钥的UUID。

——pLoadKeyInfo　　保护操作密钥信息。

输出参数描述：

——pLoadKeyInfo　　保护操作密钥信息。

——phKeyTCSI　　被加载密钥的句柄。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.2.3 Tcsip_GetPubKey

功能描述：

获取一个已经载入到TCM中的非对称密钥的公钥部分，判断给定的密钥句柄指向的密钥所绑定的平台配置信息是否与平台当前配置信息一致。

接口定义：

```
TSM_RESULT Tcsip_GetPubKey
(
      TCS_CONTEXT_HANDLE        hContext,        // in
      TCS_KEY_HANDLE            hKey,            // in
      TCM_AUTH*                 pAuth,           // in, out
      UINT32*                   pcPubKeySize,    // out
      BYTE**                    prgbPubKey       // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄

——hKey　　密钥句柄

——pAuth　　密钥的授权数据验证码

输出参数描述：

——pAuth　　密钥的授权数据验证码

——pcPubKeySize　　密钥公钥信息的长度

——prgbPubKey　　密钥公钥信息

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.2.4 Tcsip_WrapKey

功能描述：

导入一个由外部生成的 TCM_KEY 结构密钥，并指定其保护操作密钥。

接口定义：

```
TSM_RESULT Tcsip_CreateWrapKey
(
```

```
    TCS_CONTEXT_HANDLE  hContext,           // in
    TCS_KEY_HANDLE      hWrappingKey,       // in
    TCM_ENCAUTH         KeyUsageAuth,       // in
    TCM_ENCAUTH         KeyMigrationAuth,   // in
    UINT32              keyInfoSize,        // in
    BYTE*               keyInfo,            // in
    TCM_AUTH*           pAuth,              // in, out
    UINT32*             keyDataSize,        // out
    BYTE**              keyData,            // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄

——hWrappingKey　　保护操作密钥句柄

——KeyUsageAuth　　加密的使用授权数据

——KeyMigrationAuth　　加密的迁移授权数据

——keyInfoSize　　密钥信息的长度

——keyInfo　　密钥信息

——pAuth　　保护操作密钥授权数据验证码

输出参数描述：

——pAuth　　保护操作密钥授权数据验证码。

——keyDataSize　　密钥数据的长度。

——keyData　　密钥数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.2.5 Tcsip_CertifyKey

功能描述：

该命令使用一个密钥来验证另外一个密钥。

接口定义：

```
TSM_RESULT Tcsip_CertifyKey
(
  TCS_CONTEXT_HANDLE    hContext,         // in
  TCS_KEY_HANDLE        certHandle,       // in
  TCS_KEY_HANDLE        keyHandle,        // in
  TCM_NONCE             antiReplay,       // in
  TCM_AUTH*             certAuth,         // in, out
  TCM_AUTH*             keyAuth,          // in, out
  UINT32*               CertifyInfoSize,  // out
  BYTE**                CertifyInfo,      // out
  UINT32*               outDataSize,      // out
  BYTE**                outData           // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——certHandle　　验证密钥句柄。

——keyHandle　　待验证密钥句柄。

——antiReplay　　抗重放数据。

——certAuth　　验证密钥的授权数据验证码。

——keyAuth　　待验证密钥的授权数据验证码。

输出参数描述：

——certAuth　　验证密钥的授权数据验证码。

——keyAuth　　待验证密钥的授权数据验证码。

——CertifyInfoSize　　验证信息的长度。

——CertifyInfo　　验证信息。

——outDataSize　　验证信息签名值的长度。

——outData　　验证信息签名值。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.2.6 Tcsip_FlushSpecific

功能描述：

将指定的hKey从TCM中逐出。

接口定义：

```
TSM_RESULT Tcsip_FlushSpecific
(
    TCS_CONTEXT_HANDLE      hContext,    // in
    TCS_KEY_HANDLE          hKey         // in
);
```

输入参数描述：

——hContext　　TCS上下文句柄。

——hKey　　将要收回的密钥句柄（TCS端密钥句柄）。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_INVALID_CONTEXTHANDLE

TCS_E_FAIL

7.4.3　密钥协商

7.4.3.1 Tcsip_CreateKeyExchange

功能描述：

密钥协商双方A与B使用这个函数生成临时点分别为Ra,Rb。用户A将Ra传送给用户B，用户B将Rb传送给用户A。

接口定义：

```
TSM_RESULT Tcsip_CreateKeyExchange
(
    TCS_CONTEXT_HANDLE      hContext,           //in
    TSM_AUTH*               ownerAuth,          // in, out
    TCM_EXCHANGE_HANLDE *   phExchange          // out
    UINT32*                 cRxSize,            //out
    BYTE**                  prgbRxPoint         //out
);
```

输入参数描述：

——hContext　创建的上下文对象句柄。

——ownerAuth　所有者授权数据验证码。

输出参数描述：

——ownerAuth　所有者授权数据验证码。

——phExchange　协商会话句柄

——pcRxSize　指向参数 prgbRxPoint 数据长度的指针。

——prgbRxPoint　指向用户的作为临时密钥的公钥部分的指针，临时的 ECC 曲线上的点，按照国标密码算法规定的未压缩编码形式的字符串。

返回值：

TCS_SUCCESS

TCS_E_FAIL

7.4.3.2 Tcsip_GetKeyExchange

功能描述：

输入对方传送来信息，结合本地信息，协商出对称密钥以及验证码。

接口定义：

```
TSM_RESULT Tcsip_GetKeyExchange
(
    TCS_CONTEXT_HANDLE      hContext,           //in
    TCS_KEY_HANDLE          hKey,               // in
    TSM_EXCHANGE_HANLDE     hExchange,          // in
    TSM_EXCHANGE_TAG        cExchangeTag        // in
    TSM_ENCAUTH             KeyUsageAuth,       // in
    UINT32                  cPointSize,         // in
    BYTE*                   rgbPoint,           // in
    UINT32                  cRaSize             // in
    BYTE*                   rgbRa,              // in
    UINT32                  cRbSize             // in
    BYTE*                   rgbRb,              // in
    UINT32                  cRxSize             // in
    BYTE*                   rgbRx,              // in
    TSM_HKEY *              phKey               // in ,out
    TSM_AUTH*               keyAuth,            // in, out
```

```
        UINT32*                 pcSxSize            //out
        BYTE**                  prgbSxData,         // out
        UINT32*                 pcSySize            // out
        BYTE**                  prgbSyData,         // out
    );
```

输入参数描述：

——hContext　　创建的上下文对象句柄。
——hKey　　本地静态密钥句柄是已经加载的密钥句柄。
——hExchange　　协商会话句柄，TCM_CreateKeyExchange 返回的会话句柄。
——cExchangeTag　　协商标识为密钥协商的身份标识，1 代表发起方，2 代表响应方。
——KeyUsageAuth　　产生的密钥的使用授权。
——phKey　　产生的密钥结构属性，用来作为生成密钥的的存储结构。
——cPointSize　　对方静态密钥公钥信息长度。
——rgbPoint　　对方静态密钥公钥信息。
——cRaSize　　本地个人信息摘要长度。
——rgbRa　　本地个人信息摘要。
——cRbSize　　对方个人信息摘要长度。
——rgbRb　　对方个人信息摘要。
——cRxSize　　对方临时密钥公钥信息长度。
——rgbRx　　对方临时密钥公钥信息。
——keyAuth　　本地静态密钥授权验证码。

输出参数描述：

——phKey　　协商的共享密钥。
——keyAuth　　指向所有者授权数据的指针。
——pcSxSize　　用于本地验证协商过程的数据长度。
——prgbSxData　　用于本地验证协商过程的数据。
——pcSySize　　提供给对方进行验证过程的数据长度。
——prgbSyDat　　提供给对方进行验证过程的数据。

返回值：

TCS_SUCCESS
TCS_E_FAIL

7.4.3.3 Tcsip_ReleaseExchangeSession

功能描述：

该命令用来释放TCM协商过程会话。

接口定义：

```
TSM_RESULT Tcsip_ReleaseExchangeSession
(
    TCS_CONTEXT_HANDLE      hContext,   //in
    TCM_EXCHANGE_HANLDE     hExchange   // in
);
```

输入参数描述：

——hContext　　创建的上下文对象句柄。

——hExchange　　协商会话句柄。

输出参数描述：

——无。

返回值：

TCS_SUCCESS

TCS_E_FAIL

7.4.4　密钥迁移

7.4.4.1 Tcsip_AuthorizeMigrationKey

功能描述：

本函数验证授权并指定迁移方式。

接口定义：

```
TSM_RESULT Tcsip_AuthorizeMigrationKey
(
    TCS_CONTEXT_HANDLE      hContext,               // in
    TSM_MIGRATE_SCHEME      migrateScheme,          // in
    UINT32                  MigrationKeySize,       // in
    BYTE*                   MigrationKey,           // in
    TCM_AUTH*               ownerAuth,              // in, out
    UINT32*                 MigrationKeyAuthSize,   // out
    BYTE**                  MigrationKeyAuth        // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——migrateScheme　　迁移模式。

——MigrationKeySize　　迁移密钥公钥大小。

——MigrationKey　　迁移密钥公钥。

——ownerAuth　　所有者授权会话校验码。

输出参数描述：

——ownerAuth　　所有者授权会话校验码。

——MigrationKeyAuthSize　迁移认证数据大小。

——MigrationKeyAuth　　迁移认证数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.4.2 Tcsip_CreateMigrationBlob

功能描述：

本函数创建迁移数据块。

接口定义：

```
TSM_RESULT Tcsip_CreateMigrationBlob
(
```

```
    TCS_CONTEXT_HANDLE  hContext,               // in
    TCS_KEY_HANDLE      parentHandle,           // in
    TSM_MIGRATE_SCHEME  migrationType,          // in
    UINT32              MigrationKeyAuthSize,   // in
    BYTE*               MigrationKeyAuth,       // in
    UINT32              encDataSize,            // in
    BYTE*               encData,                // in
    TCM_AUTH*           arentAuth,              // in, out
    TCM_AUTH*           entityAuth,             // in, out
    UINT32*             SymEncDataSize,         // out
    BYTE**              SymEncData,             // out
    UINT32*             outDataSize,            // out
    BYTE**              outData                 // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。
——parentHandle　　待迁移密钥父密钥句柄。
——migrationType　　迁移模式。
——MigrationKeyAuthSize　　迁移认证数据大小。
——MigrationKeyAuth　　迁移认证数据。
——encDataSize　　待迁移的密钥数据大小。
——encData　　待迁移的密钥数据。
——parentAuth　　待迁移密钥父密钥授权会话校验码。
——entityAuth　　待迁移密钥迁移授权会话验证码。

输出参数描述：

——parentAuth　　待迁移密钥父密钥授权会话验证码。
——entityAuth　　待迁移密钥迁移授权会话验证码。
——SymEncDataSize　　用对称密钥加密的待迁移密钥大小
——SymEncData　　用对称密钥加密的待迁移密钥
——outDataSize　　用迁移公钥加密的待迁移密钥大小。
——outData　　用迁移公钥加密的待迁移密钥。

返回参数：

TCS_SUCCESS
TCS_E_FAIL

7.4.4.3 Tcsip_ConvertMigrationBlob

功能描述：

本函数将迁移块转换为可以被LoadKey命令使用的加密块。

接口定义：

```
TSM_RESULT Tcsip_ConvertMigrationBlob
(
  TCS_CONTEXT_HANDLE      hContext,               // in
```

```
    TCS_KEY_HANDLE          parentHandle,           // in
    TCS_KEY_HANDLE          MEKHandle,              // in
    UINT32*                 prgbMigratedDataSize,   // in
    BYTE**                  prgbMigratedData,       // in
    UINT32*                 pulEncSymKeySize,       // in
    BYTE**                  prgbEncSymKey,          // in
    TCM_AUTH*               MEKAuth,                // in, out
    TCM_AUTH*               parentAuth,             // in, out
    UINT32*                 outDataSize,            // out
    BYTE**                  outData                 // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。
——parentHandle　　已加载的父密钥句柄。
——MEKHandle　　已加载的迁移密钥句柄。
——prgbMigratedDataSize　　对称密钥加密的待迁移密钥长度。
——prgbMigratedData　　对称密钥加密的待迁移密钥。
——pulEncSymKeySize　　迁移密钥加密的对称密钥长度。
——prgbEncSymKey　　迁移密钥加密的对称密钥。
——MEKAuth　　迁移密钥授权会话验证信息。
——parentAuth　　父密钥授权会话验证信息。

输出参数描述：

——MEKAuth　　迁移密钥授权会话验证信息。
——parentAuth　　父密钥授权会话验证信息。
——outData Size　　导入的 PEK 长度。
——outData　　导入的 PEK。

返回参数：

TCS_SUCCESS
TCS_E_FAIL

7.4.5　密码学服务

7.4.5.1 Tcsip_Sign

功能描述：

利用指定的密钥执行数字签名操作并返回其数字签名。参照国标ECC算法指定的签名算法。

接口定义：

```
TSM_RESULT Tcsip_Sign
(
    TCS_CONTEXT_HANDLE      hContext,           // in
    TCS_KEY_HANDLE          keyHandle,          // in
    UINT32                  areaToSignSize,     // in
    BYTE*                   areaToSign,         // in
    TCM_AUTH*               privAuth,           // in, out
```

```
        UINT32*                 sigSize,            // out
        BYTE**                  sig                 // out
    );
```

输入参数描述：

——hContext　　创建的上下文对象句柄。

——keyHandle　　签名密钥句柄。

——areaToSignSize　待签名数据长度。

——areaToSig　　待签名数据。

——privAuth　　签名密钥授权数据验证码。

输出参数描述：

——privAuth　　授权使用密钥句柄的授权摘要。

——sigSize　　返回的数字签名数据的长度。

——sig　　最终数字签名。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.5.2 Tcsip_SM4Encrypt

功能描述：

利用已经加载的密钥使用SM4算法对输入数据进行对称加密。

接口定义：

```
TSM_RESULT Tcsip_SM4Encrypt
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    TCS_KEY_HANDLE              enckeyHandle,   // in
    BYTE*                       IV,             // in
    UINT32                      inDataSize,     // in
    BYTE*                       inData,         // in
    TCM_AUTH*                   pEncAuth,       // in, out
    UINT32                      outDataSize,    // out
    BYTE*                       outData,        // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄

——enckeyHandle　加密密钥的授权会话句柄

——IV　　CBC模式加密时使用的IV，长度为固定16字节

——inDataSize　　待加密数据长度

——inData　　待加密数据

——pEncAuth　　加密密钥的授权数据验证码

输出参数描述：

——pEncAuth　　输出的加密密钥的授权数据验证码

——outDataSize　　已加密数据长度

——outData　　已加密数据

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.5.3 Tcsip_SM4Decrypt

功能描述：

利用已经加载的密钥使用SM4算法对输入数据进行对称解密。

接口定义：

```
TSM_RESULT Tcsip_SM4Decrypt
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCS_KEY_HANDLE          DnckeyHandle,   // in
    BYTE*                   IV,             // in
    UINT32                  inDataSize,     // in
    BYTE*                   inData,         // in
    TCM_AUTH*               pEncAuth,       // in, out
    UINT32                  outDataSize,    // out
    BYTE*                   outData,        // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——DnckeyHandle　　解密密钥的授权会话句柄。

——IV　　CBC模式加密时使用的IV，长度为固定16字节

——inDataSize　　待解密数据长度。

——inData　　待解密数据。

——pEncAuth　　解密密钥的授权数据验证码。

输出参数描述：

——pEncAuth　　输出的解密密钥的授权数据验证码。

——outDataSize　　已解密数据长度。

——outData　　已解密数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.5.4 Tcsip_SM2Decrypt

功能描述：

利用指定的非对称密钥执行SM2解密操作并返回其解密结果。

接口定义：

```
TSM_RESULT Tcsip_SM2Decrypt
(
    TCS_CONTEXT_HANDLE      hContext,           // in
```

```
        TCS_KEY_HANDLE          keyHandle,          // in
        UINT32                  inDataSize,         // in
        BYTE*                   inData,             // in
        TCM_AUTH*               privAuth,           // in, out
        UINT32*                 outDataSize,        // out
        BYTE**                  outData             // out
    );
```

输入参数描述：

——hContext　创建的上下文对象句柄。

——keyHandle　解密密钥的密钥句柄。

——inDataSize　待解密数据长度。

——inData　待解密数据。

——privAuth　解密密钥的授权数据验证码。

输出参数描述：

——privAuth　解密密钥的授权数据验证码。

——outDataSize 解密后数据长度。

——outData　解密后数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.5.5 Tcsip_GetRandom

功能描述：

获取指定长度的随机数据。

接口定义：

```
    TSM_RESULT Tcsip_GetRandom
    (
        TCS_CONTEXT_HANDLE      hContext,   // in
        UINT32*                 dataSize,   // in , out
        BYTE**                  outData     // out
    );
```

输入参数描述：

——hContext 上下文对象句柄。

——dataSize 随机数据长度。

输出参数描述：

——dataSize 返回的随机数据长度。

——outData　返回的随机数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

TCS_E_NOTIMPL

7.4.6　传输会话

7.4.6.1 Tcsip_EstablishTransport

功能描述：

建立传输会话。

接口定义：

```
TSM_RESULT Tcsip_EstablishTransport
(
    TCS_CONTEXT_HANDLE      hContext,                   // in
    UINT32                  ulTransControlFlags,        // in
    TCS_KEY_HANDLE          hEncKey,                    // in
    UINT32                  ulTransSessionInfoSize,     // in
    BYTE*                   rgbTransSessionInfo,        // in
    UINT32                  ulSecretSize,               // in
    BYTE*                   rgbSecret,                  // in
    TCM_AUTH*               pEncKeyAuth,                // in, out
    TCM_LOCALITY_MOD*       pbLocality,                 // out
    TCS_HANDLE*             hTransSession,              // out
    UINT32*                 ulCurrentTicks,             // out
    BYTE**                  prgbCurrentTicks,           // out
    UINT32*                 ulTransSeq                  // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄

——ulTransControlFlags　　TCS中控制传输会话处理的标志

——hEncKey　　传输保护密钥句柄

——ulTransSessionInfoSize　　传输描述信息的长度

——rgbTransSessionInfo　　传输描述信息

——ulSecretSize　　加密的临时会话密钥长度

——rgbSecret　　加密的临时会话密钥

——pEncKeyAuth　　传输保护密钥授权数据验证码

输出参数描述：

——pEncKeyAuth　　传输保护密钥授权数据验证码

——pbLocality　　locality值

——hTransSession　　传输会话句柄

——ulCurrentTicks　　当前时钟节拍值的长度

——prgbCurrentTicks　　当前时钟节拍值

——ulTransSeq　　序列号

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.6.2 Tcsip_ExecuteTransport

功能描述：

负责命令的传输保护。

接口定义：

```
TSM_RESULT Tcsip_ExecuteTransport
(
    TCS_CONTEXT_HANDLE          hContext,                   //in
    TCM_COMMAND_CODE            unWrappedCommandOrdinal,    // in
    UINT32                      ulWrappedCmdDataInSize,     // in
    BYTE*                       rgbWrappedCmdDataIn,        // in
    UINT32*                     pulHandleListSize,          // in, out
    TCS_HANDLE**                rghHandles,                 // in, out
    TSM_AUTH*                   pWrappedCmdAuth1,           // in, out
    TSM_AUTH*                   pWrappedCmdAuth2,           // in, out
    TSM_AUTH*                   pTransAuth,                 // in, out
    UINT64*                     punCurrentTicks,            // out
    TCPA_LOCALITY_MOD*          pbLocality,                 // out
    TSM_RESULT*                 pulWrappedCmdReturnCode,    // out
    UINT32*                     ulWrappedCmdDataOutSize,    // out
    BYTE**                      rgbWrappedCmdDataOut        // out
);
```

输入参数描述：

——hContext	上下文对象的句柄。
——unWrappedCommandOrdinal	TCM操作的Ordinal
——ulWrappedCmdDataInSize	受保护命令数据的长度。
——rgbWrappedCmdDataIn	受保护命令数据。
——pulHandleListSize	句柄列表的大小
——rghHandles	TSS句柄列表
——pWrappedCmdAuth1	第一个授权会话数据。如果为 NULL，不需要授权
——pWrappedCmdAuth2	第二个授权会话数据。如果为 NULL，不需要授权
——pTransAuth	传输保护密钥授权数据验证码。

输出参数描述：

——pulHandleListSize	句柄列表的大小
——rghHandles	TSS 句柄列表
——pWrappedCmdAuth1	第一个授权会话数据。如果为 NULL，不需要授权
——pWrappedCmdAuth2	第二个授权会话数据。如果为 NULL，不需要授权
——pTransAuth	传输保护密钥授权数据验证码
——ulCurrentTicks	当前时钟节拍值的长度
——pbLocality	locality值
——pulWrappedCmdReturnCode	命令返回码
——ulWrappedCmdDataOutSize	命令响应数据大小
——rgbWrappedCmdDataOut	命令响应数据

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.6.3 Tcsip_Releasetransport

功能描述：

释放与指定的传输会话相关的资源，结束传输会话。

接口定义：

```
TSM_RESULT Tcsip_ReleaseTransport
(
    TCS_CONTEXT_HANDLE      hContext,               // in
    TCM_AUTH*               pTransAuth,             // in, out
    TCM_LOCALITY_MOD*       pbLocality,             // out
    UINT32*                 pulCurrentTicks,        // out
    BYTE**                  prgbCurrentTicks        // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——pTransAuth　　传输保护密钥授权数据验证码。

输出参数描述：

——pTransAuth　　传输保护密钥授权数据验证码。

——pbLocality　　locality值。

——pulCurrentTicks　　当前时钟节拍值的长度。

——prgbCurrentTicks　　当前时钟节拍值。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.4.7　授权协议

7.4.7.1 Tcsip_APCreate

功能描述：

建立授权会话。

接口定义：

```
TSM_RESULT Tcsip_APCreate
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCM_ENTITY_TYPE         entityType,     // in
    UINT32                  entityValue,    // in
    TCM_NONCE               callerNonce,    // in
    TCM_AUTHDATA*           pAuth,          // in, out
    TCS_AUTHHANDLE*         authHandle,     // out
    TCM_NONCE*              TcmNonce,       // out
    UINT32 *                AntiReplaySeq   // out
);
```

输入参数描述：

——hContext　　上下文对象的句柄。

——entityType　　实体类型。
——entityValue　　实体值。
——callerNonce　　随机数。
——pAuth　　实体授权数据验证码。
输出参数描述：
——pAuth　　实体授权数据验证码。
——authHandle　　授权会话句柄。
——TcmNonce　　随机数。
——AntiReplaySeq　　序列号。
返回参数：
TCS_SUCCESS
TCS_E_FAIL

7.4.7.2 Tcsip_APTerminate

功能描述：
该命令终止授权协议，释放指定的授权会话及相关资源。
接口定义：

```
TSM_RESULT Tcsip_APTerminate
(
    TCS_CONTEXT_HANDLE      hContext,    // in
    TCS_AUTHHANDLE*         authHandle, // in
    TCM_AUTH*               pAuth        // in
);
```

输入参数描述：
——hContext　　上下文对象的句柄。
——authHandle　　实体授权会话句柄。
——pAuth　　实体授权数据验证码。
返回参数：
TCS_SUCCESS
TCS_E_FAIL

7.5　完整性度量与报告

7.5.1　平台配置寄存器管理

7.5.1.1 Tcsip_Extend

功能描述：
在 PCR 寄存器中增加一个度量值。
接口定义：

```
TSM_RESULT Tcsip_Extend
(
    TCS_CONTEXT_HANDLE      hContext,    // in
    TCM_PCRINDEX            pcrNum,      // in
    TCM_DIGEST              inDigest,    // in
```

```
        TCM_PCRVALUE*           outDigest   // in, out
    );
```

输入参数描述：

——hContext　上下文对象的句柄。

——pcrNum　PCR索引。

——inDigest　被度量部件的特征数据的256比特位的哈希值。

——outDigest　指向新的度量值的指针。

输出参数描述：

——outDigest 新的度量值。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.5.1.2 Tcsip_PcrRead

功能描述：

读取指定PCR寄存器的当前值。

接口定义：

```
TSM_RESULT Tcsip_PcrRead
(
    TCS_CONTEXT_HANDLE      hContext,   // in
    TCM_PCRINDEX            pcrNum,     // in
    TCM_PCRVALUE*           outDigest   // out
);
```

输入参数描述：

——hContext　上下文对象的句柄。

——pcrNum　PCR索引。

输出参数描述：

——outDigest　指定PCR寄存器的当前值。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.5.1.3 Tcsip_Quote

功能描述：

对指定的PCR值，返回给定密钥的签名。

接口定义：

```
TSM_RESULT Tcsip_Quote
(
    TCS_CONTEXT_HANDLE      hContext,       // in
    TCS_KEY_HANDLE          keyHandle,      // in
    TCM_NONCE               antiReplay,     // in
    UINT32                  pcrTargetSize,  // in
    BYTE*                   pcrTarget,      // in
```

```
        TCM_AUTH*                   privAuth,       // in, out
        UINT32*                     pcrDataSize,    // out
        BYTE**                      pcrData,        // out
        UINT32*                     sigSize,        // out
        BYTE**                      sig             // out
    );
```

输入参数描述：

——hContext　　上下文对象的句柄。

——keyHandle　　密钥句柄。

——antiReplay　　防重放攻击数据。

——pcrTargetSize　　目标PCR长度。

——pcrTarget　　目标PCR。

——privAuth　　密钥授权验证码。

输出参数描述：

——privAuth　　密钥授权验证码。

——pcrDataSize　　PCR数据长度

——pcrData　　PCR数据。

——sigSize　　签名数据长度。

——sig　　签名数据。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

7.5.1.4 Tcsip_PcrReset

功能描述：

对于可重置的PCR，设置为启动时的初始值。

接口定义：

```
TSM_RESULT Tcsip_PcrReset
(
    TCS_CONTEXT_HANDLE          hContext,       // in
    UINT32                      pcrTargetSize,  // in
    BYTE*                       pcrTarget       // in
 );
```

输入参数描述：

——hContext　　上下文对象句柄。

——pcrTargetSize　　目标PCR长度。

——pcrTarget　　目标PCR。

输出参数描述：

——无。

返回参数：

TCS_SUCCESS

TCS_E_FAIL

TCS_E_NOTIMPL

8 TDDL 设备驱动库

8.1 TDDL 架构

TDDL是存在于TSM和TCM设备驱动(TDD)之间的模块。TDDL实现TCM核心服务(TCS)与TCM间的数据通信，将TCM内核模式的驱动程序(TDD)转化为用户模式执行。

TDDL 提供的接口为 TCM 设备驱动库接口(TDDLI)。TCM 制造商必须定义这个设备驱动库与 TCM 设备之间的接口。在定义接口时，制造商可以选择不同的机制，来实现 TDDL 与任意一个内核模式 TCM 驱动或者 TCM 软件模拟器之间的数据通信和资源配置。

8.2 TDDL 内存管理

TDDL 使用传统的内存分配方法，即被调用的应用程序为每个接口的输入输出参数分配内存。相应地，在调用 TDDL 之后，应用程序还负责将使用完的内存释放掉。

8.3 TDDL 错误码与定义

表26列出了TDDL出现的错误码。另外，每个Tddli函数定义中都会列出与其有关的错误码（具体数值可由厂商自行定义具体值)。

表 26 TDDL 错误码定义表

类型	值	定义
TDDL_SUCCESS	0x00	操作成功完成
TDDL_E_FAIL	0x01	操作失败
TDDL_E_BAD_PARAMETER	0x02	一个或多个参数错误
TDDL_E_OUTOFMEMORY	0x03	内存不足
TDDL_E_TIMEOUT	0x04	操作超时
TDDL_E_ALREADY_OPENED	0x81	TCM 设备驱动已打开
TDDL_E_ALREADY_CLOSED	0x82	TCM 设备驱动已关闭
TDDL_E_INSUFFICIENT_BUFFER	0x83	缓冲区空间太小
TDDL_E_COMMAND_COMPLETED	0x84	命令已完成
TDLL_E_COMMAND_ABORTED	0x85	TCM 执行命令失败
TDDL_E_IOERROR	0x86	向 TCM 传输是产生一个 IO 错误
TDDL_E_BADTAG	0x87	域或子域的标识符错误或不支持
TDDL_E_COMPONENT_NOT_FOUND	0x88	TCM 设备驱动未运行

8.4 TDDL 接口

8.4.1 Tddli_Open

功能描述

该函数与TCM设备驱动建立连接。实现该函数后，TCM设备驱动必须准备好执行应用程序所需的TCM指令。TDDL确保应用程序通过唯一的入口访问TCM设备。如果该函数调用失败，表明TCM设备驱动可能没有装载成功、没有正常启动或者TCM不支持任何受保护的请求。

该函数必须在Tddli_GetStatus，Tddli_GetCapability，Tddli_SetCapability和Tddli_TransmitData函数之前调用。

接口定义

TSM_RESULT Tddli_Open();

输入参数描述：

——无

输出参数描述：

——无

返回值：

TDDL_SUCCESS

TDDL_E_FAIL

TDDL_E_ALREADY_OPENED

TDDL_E_TIMEOUT

TDDL_E_COMPONENT_NOT_FOUND

8.4.2 Tddli_Close

功能描述

该函数与TCM设备驱动断开连接。实现该函数后，TCM设备驱动将清除所有用来与TDDL进行连接的资源。如果该函数调用失败，表明TCM设备驱动未彻底清除资源，或者需要重新启动或装载。

接口定义

TSM_RESULT Tddli_Close();

输入参数描述：

——无

输出参数描述：

——无

返回值：

TDDL_SUCCESS

TDDL_E_FAIL

TDDL_E_ALREADY_CLOSED

TDDL_E_TIMEOUT

TDDL_E_COMPONENT_NOT_FOUND

8.4.3 Tddli_Cancel

功能描述

该函数用于删除未完成的TCM命令。应用程序可以在独立的上下文中调用这个函数，来中断一个未完成的TCM命令。这里的TCM命令必须由调用Tddli_TransmitData函数得到。

如果TCM设备驱动没有得到先前TCM命令的返回值，则其必须响应该函数，并在进程调用时返回

TDDL_COMMAND_ABORTED。

接口定义

```
TSM_RESULT Tddli_Cancel( );
```

输入参数描述：

——无

输出参数描述：

——无

返回值

```
TDDL_SUCCESS
TDDL_E_FAIL
TDDL_E_COMPONENT_NOT_FOUND
TDDL_E_TIMEOUT
TDDL_E_ALREADY_CLOSED
```

8.4.4 Tddli_GetCapability

功能描述

该函数获得TCM硬件、固件或者设备驱动的特征参数，比如固件版本、驱动版本等。

接口定义

```
TSM_RESULT Tddli_GetCapability
(
    UINT32 CapArea,          // in
    UINT32 SubCap,           // in
    BYTE* pCapBuf,           // out
    UINT32* pCapBufLen       // in, out
);
```

输入参数描述

——CapArea　　域类型标识属性

——SubCap　　要获得的域类型子属性

——pCapBuf　　缓冲区字节

——pCapBufLen　　缓冲区实际存放数据的字节数

输入参数的属性和子属性见表 27

输出参数描述

——pCapBuf　　指向缓冲区的指针，该缓冲区用于存放获得的特征参数。

——pCapBufLen　　缓冲区实际存放数据的字节数

返回值：

```
TDDL_SUCCESS
TDDL_E_BAD_PARAMETER
TDDL_E_OUTOFMEMORY
TDDL_E_BADTAG
TDDL_E_FAIL
```

表 27 属性定义

属性	子属性	描述
TCM_CAP_VERSION	TSM_CAP_PROP_DRV	返回TCM设备驱动的版本值 该值为 TCM_VERSION格式
TCM_CAP_VERSION	TSM_CAP_PROP_FW	返回当前TCM固件的版本值 该值为 TCM_VERSION格式
TCM_CAP_VERSION	TSM_CAP_PROP_FW_DATE	返回固件的发布日期该日期为3字节 格式：mm/dd/yy (mm=month, dd=day, yy=year) 1月～12月对应01～12
TCM_CAP_PROPERTY	TCM_CAP_PROP_MANUFA CTURER	返回驱动制造商的名称 该数据为不带结尾null值的ASCII串
TCM_CAP_PROPERTY	TSM_CAP_PROP_MODULE_TYPE	返回制造商指定的驱动类型名 该数据为不带结尾null值的ASCII串
TCM_CAP_PROPERTY	TSM_CAP_PROP_GLOBAL_STATE	返回模块的全局状态，比如已初始化等
TCM_CAP_VENDOR	TCM_CAP_VENDOR_XXX	返回制造商指定的TCM域

8.4.5 Tddli_SetCapability

功能描述

该函数用于设置TCM硬件、固件或者设备驱动的特征参数。应用程序设置的TCM设备驱动或操作参数可能是由TCM制造商定义的。

接口定义

```
TSM_RESULT Tddli_SetCapability
(
    UINT32 CapArea,      // in
    UINT32 SubCap,       // in
    BYTE*  pSetCapBuf,   // in
    UINT32 SetCapBufLen // in
);
```

输入参数描述

——CapArea　　域类型标识，这个函数设置该域的参数

——SubCap　　域类型子码，这个函数设置该域的参数

——pSetCapBuf　　指向缓冲区的指针，该缓冲区用于存放设置的参数

——SetCapBufLen　　所需缓冲区的字节大小

输出参数描述

无

返回值：

```
TDDL_SUCCESS
TDDL_E_OUTOFMEMORY
TDDL_E_BAD_PARAMETER
TDDL_E_BADTAG
```

TDDL_E_FAIL

8.4.6 Tddli_GetStatus

功能描述

该函数获得TCM驱动和设备的状态。应用程序可以利用这个函数衡量TCM子系统的安全性。

接口定义

```
TSM_RESULT Tddli_GetStatus
(
    UINT32 ReqStatusType, // in
    UINT32* pStatus       // out
);
```

输入参数描述

——ReqStatusType　所需的状态值类型，驱动或设备

输出参数描述

——pStatus　获得的状态值。

输出参数的状态值见表 28.

返回值：

TDDL_SUCCESS

TDDL_E_BAD_PARAMETER

TDDL_E_INSUFFICIENT_BUFFER

TDDL_E_FAIL

表 28 状态类型定义

状态类型定义	输出状态值	描述
TDDL_DRIVER_STATUS	TDDL_DRIVER_OK	TCM驱动正常
TDDL_DRIVER_STATUS	TDDL_DRIVER_FAILED	TCM驱动工作不正常
TDDL_DRIVER_STATUS	TDDL_DRIVER_NOT_OPENED	设备已找到，但无法打开相应的驱动程序
TDDL_DEVICE_STATUS	TDDL_DEVICE_OK	TCM设备正常
TDDL_DEVICE_STATUS	TDDL_DEVICE_UNRECOVERABLE	TCM设备存在一个不可恢复的错误
TDDL_DEVICE_STATUS	TDDL_DEVICE_RECOVERABLE	TCM设备存在一个可恢复的错误
TDDL_DEVICE_STATUS	TDDL_DEVICE_NOT_FOUND	TCM设备未找到

8.4.7 Tddli_TransmitData

功能描述

该函数直接将TCM命令传给TCM设备驱动，使TCM完成相应操作。同时为TCM参数提供通道。

接口定义：

```
TSM_RESULT Tddli_TransmitData
(
    BYTE*       pTransmitBuf,       // in
    UINT32      TransmitBufLen,     // in
    BYTE*       pRececeiveBuf,      // out
    UINT32*     pRececeiveBufLen    // out
);
```

输入参数描述

——pTransmitBuf　　指向缓冲区的指针，该缓冲区用于存放 TCM 传输的数据

——TransmitBufLen　TCM 传输数据的字节数

输出参数描述

——pRececeiveBuf　　指向缓冲区的指针，该缓冲区用于存放 TCM 得到的数据

——pRececeiveBufLen　　得到数据的字节数

返回值：

TDDL_SUCCESS

TDDL_E_INSUFFICIENT_BUFFER

TDDL_E_IOERROR

TDDL_E_FAIL

附 录 A
（规范性附录）
接口数据结构

A.1 基础定义

A.1.1 数据类型

A.1.1.1 指针和句柄长度

指针和句柄长度均为 32 位。

A.1.1.2 基本类型

基本类型定义见表 A.1。

表 A.1 基本类型定义表

类型	定义
UINT32	无符号 32 位整型
BYTE	无符号字符型，1 字节
TSM_BOOL	带符号字符型，1 字节
TSM_UNICODE	16 比特的序列(比特串)
PVOID	32 位指针

A.1.1.3 布尔类型

布尔类型定义见表 A.2。

表 A.2 布尔类型定义表

名称	值	描述
TRUE	0x01	真
FALSE	0x00	假

A.1.1.4 导出类型

导出类型定义见表 A.3。

表 A.3 导出类型定义表

类型	定义	描述
TSM_FLAG	UINT32	对象属性
TSM_HOBJECT	UINT32	基本对象句柄
TSM_ALGORITHM_ID	UINT32	TSM 算法标
TSM_MIGRATE_SCHEME	UINT32	TSM 迁移方案标识

类型	定义	描述
TSM_KEY_USAGE_ID	UINT32	TSM 密钥使用类型标识
TSM_KEY_ENC_SCHEME	UINT32	TSM 加密方案标识
TSM_KEY_SIG_SCHEME	UINT32	TSM 签名方案标识
TSM_EVENTTYPE	UINT32	TSM 事件类型
TSM_COUNTER_ID	UINT32	计数器标识
TSM_RESULT	UINT32	TSM 接口命令结果

A.1.1.5 对象类型

对象类型定义见表 A.4。

表 A.4　对象类型定义表

类型	定义	描述
TSM_HCONTEXT	TSM_HOBJECT	上下文对象句柄
TSM_HPOLICY	TSM_HOBJECT	安全策略对象句柄
TSM_HTCM	TSM_HOBJECT	TCM 对象句柄
TSM_HKEY	TSM_HOBJECT	密钥对象句柄
TSM_HENCDATA	TSM_HOBJECT	加密数据对象句柄
TSM_HPCRS	TSM_HOBJECT	PCR 组合对象句柄
TSM_HHASH	TSM_HOBJECT	杂凑对象句柄.
TSM_HNVSTORE	TSM_HOBJECT	NV 数据对象句柄
TSM_HMIGDATA	TSM_HOBJECT	迁移数据处理对象句柄
TSM_HEXCHANGE	TSM_HOBJECT	密钥协商对象句柄

A.1.2　定义的常量

A.1.2.1 对象类型定义

对象类型的定义与 Tspi_Context_CreateObject 方法一起使用。定义的对象类型基于数据类型 TSM_FLAG。对象类型定义见表 A.5。

表 A.5　对象类型定义表

对象类型	描述
TSM_OBJECT_TYPE_POLICY	策略对象
TSM_OBJECT_TYPE_KEY	密钥对象(包括对称与非对称)
TSM_OBJECT_TYPE_ENCDATA	加密数据对象；限定使用范围的数据、密封数据或信封封装数据
TSM_OBJECT_TYPE_PCRS	PCR 对象
TSM_OBJECT_TYPE_HASH	杂凑对象
TSM_OBJECT_TYPE_NVSTORE	非易失性存储对象
TSM_OBJECT_TYPE_MIGDATA	迁移数据处理对象
TSM_OBJECT_TYPE_EXCHANGE	密钥协商对象

A.1.2.2 对象初始化定义

对象初始化标记的定义与Tspi_Context_CreateObject方法一起使用。定义初始化标记基于数据类型TSM_FLAG。对象初始化标记定义见表A.6。

表A.6　对象初始化定义表

初始化标记	描述
TSM_KEY_SIZE_DEFAULT_SYM	缺省的对称密钥长度
TSM_KEY_SIZE_DEFAULT_ASY	缺省的非对称密钥长度
TSM_SM4KEY_SIZE_128	SM4的密钥长度为128-bit
TSM_SM2KEY_SIZE_256	SM2的私钥长度为256-bit
TSM_SM2KEY_SIZE_512	SM2的公钥长度为512-bit
TSM_SM2KEY_TYPE_STORAGE	SM2存储加密密钥
TSM_SM2KEY_TYPE_SIGNING	SM2签名密钥
TSM_SM2KEY_TYPE_BIND	SM2加密密钥
TSM_SM2KEY_TYPE_IDENTITY	SM2身份标识密钥
TSM_SM2KEY_TYPE_AUTHCHANGE	临时性SM2密钥，用于改变授权数据值
TSM_SM2KEY_TYPE_MIGRATE	迁移保护密钥
TSM_SM4KEY_TYPE_STORAGE	SM4存储加密密钥
TSM_SM4KEY_TYPE_BIND	SM4加密密钥
TSM_KEY_NON_VOLATILE	非易失性密钥，启动时可以不加载
TSM_KEY_VOLATILE	易失性密钥，启动时必须加载
TSM_KEY_NOT_MIGRATABLE	不可迁移密钥(缺省属性)
TSM_KEY_MIGRATABLE	可迁移的密钥
TSM_KEY_NO_AUTHORIZATION	无需授权的密钥(缺省属性)
TSM_KEY_AUTHORIZATION	使用需授权的密钥
TSM_KEY_AUTHORIZATION_PRIV_USE_ONLY	密钥的私钥部分使用时需授权的密钥
TSM_KEY_STRUCT_KEY	使用TCM密钥对象
TSM_KEY_EMPTY_KEY	非TCM密钥模板(空TSM密钥对象)
TSM_KEY_TSP_SMK	使用TCM SMK模板(用于SMK的TSM密钥对象)
TSM_ENCDATA_SEAL	用于数据封装操作的数据对象
TSM_ENCDATA_BIND	用于加密操作的数据对象
TSM_ENCDATA_ENVELOP	用于数字信封操作的数据对象
TSM_HASH_DEFAULT	缺省密码杂凑算法
TSM_HASH_SCH	SM3算法的杂凑对象
TSM_HASH_OTHER	其它算法的杂凑对象
TSM_POLICY_USAGE	用于授权的策略对象
TSM_POLICY_MIGRATION	用于密钥迁移的策略对象
TSM_POLICY_OPERATOR	用于操作者授权的策略对象
TSM_PCRS_STRUCT_INFO	使用TCM的PCR对象
TSM_EXCHANGE_DEFAULT	密钥协商对象

A.1.2.3 上下文对象的属性定义

上下文对象的属性标记定义见表A.7，上下文对象的属性值定义见表A.8。

表 A.7　上下文对象属性标记表

标记	描述
TSM_TSPATTRIB_CONTEXT_SILENT_MODE	获取或设置一个上下文对象的休眠模式
TSM_TSPATTRIB_CONTEXT_MACHINE_NAME	获得 TSM 的机器名
TSM_TSPATTRIB_CONTEXT_VERSION_MODE	获取或设置版本，该信息可用于处理上下文对象的模式
TSM_TSPATTRIB_CONTEXT_CONNECTION_VERSION	获得连接的最高支持版本(TSM 和 TCM 的最高通用版本)
TSM_TSPATTRIB_CONTEXT_TRANSPORT	获取或设置与该上下文对象相关联的传输会话的相关属性
TSM_TSPATTRIB_SECRET_HASH_MODE	获得/设置串的杂凑操作模式
TSM_TSPATTRIB_CONTEXTTRANS_CONTROL	打开与关闭传输会话
TSM_TSPATTRIB_CONTEXTTRANS_MODE	控制传输会话的特性
TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	获取或设置在弹出模式下杂凑操作

表 A.8 上下文对象属性值定义表

标记	描述
TSM_TSPATTRIB_CONTEXT_NOT_SILENT	请求用户提供密码时，显示 TSM 对话框
TSM_TSPATTRIB_CONTEXT_SILENT	不显示TSM对话框（默认）
TSM_TSPATTRIB_TRANSPORT_NO_DEFAULT_ENCRYPTION	使传输会话中数据加密功能关闭
TSM_TSPATTRIB_TRANSPORT_DEFAULT_ENCRYPTION	使传输会话中数据加密功能打开
TSM_TSPATTRIB_TRANSPORT_EXCLUSIVE	排它传输模式

A.1.2.4 TCM 对象属性定义

TCM 对象属性定义见表 A.9，证书部分的属性标记见表 A.10。

表 A.9　TCM 对象属性定义表

标记	描述
TSM_TSPATTRIB_TCMCAP_SET_VENDOR	允许厂商在 TCM 中按照常规受保护区域位置的需求设置特定区域
TSM_TSPATTRIB_TCM_ORDINAL_AUDIT_STATUS	向审计列表添加或清除一个命令码
TSM_TSPATTRIB_TCMCAP_MIN_COUNTER	表示单调计数器递增的最小时间间隔，该间隔以 1/10 秒为度量单位
TSM_TSPATTRIB_TCMCAP_FLAG_VOLATILE	返回 TCM 启动标志

表 A.10　证书部分的属性标记

标记	子标记	描述
TSM_TSPATTRIB_TCM_CREDENTIAL	TSM_TCMATTRIB_EKCERT	密码模块证书 blob
	TSM_TCMATTRIB_PLATF ORMCERT	平台身份证书 blob

标记	子标记	描述
TSM_TSPATTRIB_TCM_ORDINAL_AUDIT_STATUS	TCM_CAP_PROP_TCM_SET_ORDINAL_AUDIT	向审计列表中加入一个命令码
	TCM_CAP_PROP_TCM_CLEAR_ORDINAL_AUDIT	要添加到审计列表中或者要从审计列表钟删除的命令码

A.1.2.5 策略对象属性定义

策略对象属性定义见表 A.11，子属性定义见表 A.12。

表 A.11　策略对象属性定义

标记	描述
TSM_TSPATTRIB_POLICY_SECRET_LIFETIME	设置/获得授权数据的生命周期
TSM_TSPATTRIB_POLICY_POPUPSTRING	设置一个以 NULL 结尾的 TSM_UNICODE 字符串，该字符串在 TSM 策略弹出对话框中显示。
TSM_TSPATTRIB_SECRET_HASH_MODE	获得/设置上下文或策略对象的杂凑操作模式

表 A.12　子属性定义

标记	描述
TSM_TSPATTRIB_POLSECRET_LIFETIME_ALWAYS	授权数据总是有效
TSM_TSPATTRIB_POLSECRET_LIFETIME_COUNTER	授权数据可多次使用
TSM_TSPATTRIB_POLSECRET_LIFETIME_TIMER	授权数据有效期为 n 秒
TSM_TSPATTRIB_SECRET_HASH_MODE_POPUP	获得/设置弹出模式的杂凑行为

A.1.2.6 密钥对象属性定义

密钥对象属性定义见表 A.13，子属性定义见表 A.14。

表 A.13　密钥对象属性定义

标记	描述
TSM_TSPATTRIB_KEY_REGISTER	获得/设置密钥所注册的永久存储区
TSM_TSPATTRIB_KEY_BLOB	获得/设置密钥 blob
TSM_TSPATTRIB_KEY_INFO	获得密钥信息
TSM_TSPATTRIB_KEY_UUID	获得 TSM_UUID 结构，该结构包含为密钥所分配的 UUID
TSM_TSPATTRIB_KEY_PCR	获得密钥所封装到的 PCR 信息（用于采用 TSM_KEY_STRUCT_KEY 结构的密钥）
TSM_TSPATTRIB_KEY_CONTROLBIT	获得加载的密钥属性

表 A.14　子属性定义

标记	描述
TSM_TSPATTRIB_KEYREGISTER_USER	密钥注册到用户永久存储区

标记	描述
TSM_TSPATTRIB_KEYREGISTER_SYSTEM	密钥注册到系统永久存储区
TSM_TSPATTRIB_KEYREGISTER_NO	密钥未注册到永久存储区
TSM_TSPATTRIB_KEYBLOB_BLOB	密钥 blob 形式的密钥信息
TSM_TSPATTRIB_KEYBLOB_PUBLIC_KEY	公钥 blob 形式的公钥信息
TSM_TSPATTRIB_KEYBLOB_PRIVATE_KEY	私钥 blob，是加密的私钥信息
TSM_TSPATTRIB_KEYINFO_SIZE	密钥的比特长度
TSM_TSPATTRIB_KEYINFO_USAGE	密钥使用信息
TSM_TSPATTRIB_KEYINFO_KEYFLAGS	密钥标志
TSM_TSPATTRIB_KEYINFO_AUTHUSAGE	密钥授权使用信息
TSM_TSPATTRIB_KEYINFO_ALGORITHM	密钥算法标识
TSM_TSPATTRIB_KEYINFO_SIGSCHEME	密钥签名方案
TSM_TSPATTRIB_KEYINFO_ENCSCHEME	密钥加密方案
TSM_TSPATTRIB_KEYINFO_MIGRATABLE	若为真则密钥是可迁移的
TSM_TSPATTRIB_KEYINFO_VOLATILE	若为真则密钥是易失性的
TSM_TSPATTRIB_KEYINFO_AUTHDATAUSAGE	若为真则需要授权
TSM_TSPATTRIB_KEYINFO_VERSION	TSM 版本结构信息
TSM_TSPATTRIB_KEYINFO_KEYSTRUCT	密钥结构类型
TSM_TSPATTRIB_KEYPCR_LOCALITY_ATCREATION	创建 blob 时的 Locality
TSM_TSPATTRIB_KEYPCR_LOCALITY_ATRELEASE	使用密钥所需要的 locality
TSM_TSPATTRIB_KEYPCR_CREATION_SELECTION	选定创建 blob 时活动的 PCR
TSM_TSPATTRIB_KEYPCR_RELEASE_SELECTION	选定使用密钥所需要的 PCR
TSM_TSPATTRIB_KEYPCR_DIGEST_ATCREATION	digestAtCreation 值
TSM_TSPATTRIB_KEYPCR_DIGEST_ATRELEASE	digestAtRelease 值

A.1.2.7 数据对象属性定义

数据对象属性定义见表 A.15，子属性定义见表 A.16。

表 A.15　数据对象属性定义

标记	描述
TSM_TSPATTRIB_ENCDATA_BLOB	获得/设置数据 blob
TSM_TSPATTRIB_ENCDATA_PCR	获得/设置用于封装数据的 PCR 信息(对于使用 TSM_PCRS_STRUCT_INFO 结构的加密数据对象)
TSM_TSPATTRIB_ENCDATA_SEAL	获得/设置封装操作的参数

表 A.16　子属性定义

标记	描述
TSM_TSPATTRIB_ENCDATABLOB_BLOB	数据 blob,表示加密的数据,取决于其类型(封装、加密或数字信封)
TSM_TSPATTRIB_ENCDATAPCR_DIGEST_ATCREATION	获得封装时的 PCR 的组合摘要值
TSM_TSPATTRIB_ENCDATAPCR_DIGEST_ATRELEASE	获得为解封装选择的 PCR 的组合摘要值

TSM_TSPATTRIB_ENCDATAPCR_SELECTION	获得表示活动 PCR 的位图
TSM_TSPATTRIB_ENCDATAPCRLONG_LOCALITY_ATCREATION	获得封装时的 Locality 值
TSM_TSPATTRIB_ENCDATAPCRLONG_LOCALITY_ATRELEASE	获得解封时的 Locality 值
TSM_TSPATTRIB_ENCDATAPCRLONG_CREATION_SELECTION	获得表示封装时活动的 PCR 的位图
TSM_TSPATTRIB_ENCDATAPCRLONG_RELEASE_SELECTION	获得表示解封时活动的 PCR 的位图
TSM_TSPATTRIB_ENCDATAPCRLONG_DIGEST_ATCREATION	获得封装时选择的 PCR 的组合摘要
TSM_TSPATTRIB_ENCDATAPCRLONG_DIGEST_ATRELEASE	获得为解封选择的 PCR 的组合摘要值

A.1.2.8 NV 对象属性定义

NV 对象属性定义见表 A.17，子属性见表 A.18，NV 常量-NV Index 位定义 见表 A.19，NV 常量- NV Index Masks 见表 A.20，NV 常量- NV Required Indexes 见表 A.21，NV 常量-NV 权限（Permissions）见表 A.22。

表 A.17 NV 对象属性定义

标记	描述
TSM_TSPATTRIB_NV_INDEX	NV 存储区的索引
TSM_TSPATTRIB_NV_PERMISSIONS	NV 权限
TSM_TSPATTRIB_NV_STATE	获得 NV 存储区的各种状态,只有在 NV 空间已经被定义的情况下才可用
TSM_TSPATTRIB_NV_DATASIZE	定义的 NV 存储区的大小
TSM_TSPATTRIB_NV_PCR	NV 存储区的 PCR 限制

表 A.18 子属性定义

标记	描述
TSM_TSPATTRIB_NVSTATE_READSTCLEAR	每次启动 TCM 时设置为 FALSE，在 datasize 为 0 的 ReadValuexxx 之后，设置为 TRUE。
TSM_TSPATTRIB_NVSTATE_WRITESTCLEAR	每次启动 TCM 时设置为 FALSE，在 datasize 为 0 的 WriteValuexxx 之后，设置为 TRUE
TSM_TSPATTRIB_NVSTATE_WRITEDEFINE	在 TCM 定义 NV 空间之后设置为 FALSE，在 datasize 为 0 的 WriteValue 的成功调用之后，设置为 TRUE
TSM_TSPATTRIB_NVPCR_READPCRSELECTION	Locality 选择掩码，用于 NV 区域的 PCR 读取限制
TSM_TSPATTRIB_NVPCR_READDIGESTATRELEASE	digestAtRelease，用于 NV 区域的 PCR 读取限制
TSM_TSPATTRIB_NVPCR_READLOCALITYATRELEASE	Locality 掩码，用于 NV 区域的 PCR 读取限制

TSM_TSPATTRIB_NVPCR_WRITEPCRSELECTION	Locality 选择掩码，用于 NV 区域的 PCR 写限制
TSM_TSPATTRIB_NVPCR_WRITEDIGESTATRELEASE	DigestAtRelease，用于 NV 区域的 PCR 写限制
TSM_TSPATTRIB_NVPCR_WRITELOCALITYATRELEASE	Locality 掩码，用于 NV 区域的 PCR 写限制

表 A.19 NV 常量-NV Index 位定义

NV Index 定义	描述
TSM_NV_TCM	值“T”，该索引为 TCM 厂商保留位 0 表示本标准定义，1 表示厂商特定值
TSM_NV_PLATFORM	值“P”，该索引为平台厂商保留位 1 表示厂商特定值
TSM_NV_USER	值“U”，该索引为平台用户保留 1 表示平台用户特定值
TSM_NV_DEFINED	值“D”，该索引已被定义 1 表示该索引被永久定义，任何定义该索引空间的操作都将失败

表 A.20 NV 常量- NV Index Masks

NV Index Masks	描述
TSM_NV_MASK_DOMAIN_BITS	位，值“1”用于索引域
TSM_NV_MASK_RESERVED	保留位
TSM_NV_MASK_PURVIEW	掩码范围
TSM_NV_MASK_INDEX	这些位用于索引

表 A.21 NV 常量- NV Required Indexes

NV Required Indexes	描述
TSM_NV_INDEX_LOCK	长度必须为 0 该值打开 NV 授权保护 一旦执行，所有 NV 存储区所定义的保护都被打开，该值从不重置
TSM_NV_INDEX0	长度必须为 0 该值允许设置永久锁定位，该位只能在 TCM 启动时重置

表 A.22 NV 常量-NV 权限（Permissions）

NV 权限（Permissions）	描述
TCM_NV_PER_READ_STCLEAR	TCM 启动后该值只能被读一次 Datasize 为 0 的读操作将设置为锁定
TCM_NV_PER_AUTHREAD	读该值需要授权
TCM_NV_PER_OWNERREAD	读该值需要 TCM 所有者授权
TCM_NV_PER_PPREAD	读该值需要物理现场
TCM_NV_PER_GLOBALLOCK	在成功地向索引 0 写之前该值都是可写的

NV 权限（Permissions）	描述
	对此属性的锁定可由 TCM 启动命令复位 锁定由 SV -> bGlobalLock 保持
TCM_NV_PER_WRITE_STCLEAR	Datasize 为 0 的指定索引是成功的 此属性的锁定由 TCM 启动命令复位 bWriteSTClear 中每个区域都保持锁定
TCM_NV_PER_WRITEDEFINE	TCM 定义 NV 空间后，该值只能被写一次 bWriteDefine 中每个区域都保持锁定，datasize 为 0 的索引写操作将设置锁定
TCM_NV_PER_WRITEALL	该值必须在一个单一的操作中被写入
TCM_NV_PER_AUTHWRITE	写该值需要授权
TCM_NV_PER_OWNERWRITE	写该值需要 TCM 所有者授权
TCM_NV_PER_PPWRITE	写该值需要物理现场

A.1.2.9 迁移数据对象属性定义

迁移数据对象属性见表 A.23。

表 A.23　迁移数据对象属性定义

属性	子属性	描述
TSM_MIGATTRIB_MIGRATION BLOB	TSM_MIGATTRIB_MIGRATION_SM4_BLOB	CreateBlob 操作输出的数据包(采用 SM4 算法)
	TSM_MIGATTRIB_MIGRATION_XOR_BLOB	CreateBlob 操作输出的数据包
	TSM_MIGATTRIB_MIGRATION_REWRAPPED_BLOB	密钥迁移操作后的数据格式
	TSM_MIGATTRIB_MIG_DESTINATION_PUBKEY_BLOB	经过认证的目标平台公钥
TSM_MIGATTRIB_PAYLOAD_TYPE	TSM_MIGATTRIB_PT_MIGRATE_RESTRICTED	本地创建的密钥
	TSM_MIGATTRIB_PT_MIGRATE_EXTERNAL	迁移到此的密钥

A.1.2.10　杂凑对象属性定义

杂凑对象属性见表 A.24。

表 A.24　杂凑对象属性定义

属性	描述
TSM_TSPATTRIB_ALG_IDENTIFIER	获得/设置密码杂凑算法标识

A.1.2.11　Secret Mode 策略定义

策略模式标记定义,可用于 Tspi_Policy_SetSecret( )方法。定义的授权数据模式基于 TSM_FLAG 数据类型，Secret Mode 策略定义见表 A.25。

表 A.25　Secret Mode 策略定义

Secret Mode	描述
TSM_SECRET_MODE_NONE	不设置策略授权信息，这与 0x00 的授权不一样
TSM_SECRET_MODE_SM3	输入经外部杂凑计算得到的 32 个字节数组作为授权数据。TSM 不会修改这个输入数据
TSM_SECRET_MODE_PLAIN	输入一个任意字节的数组，然后将其进行杂凑运算，作为授权数据
TSM_SECRET_MODE_POPUP	TSM 向用户询问一个 TSM_UNICODE 类型的授权口令串，该串需要进行杂凑运算作为授权数据

A.1.2.12 Secret 生命周期策略定义

通过 Tspi_SetAttribUint32()和 Tspi_GetAttribUint32()函数能够设置/获取授权数据的生命周期。生命周期模式的定义是基于 TSM_FLAG 类型的。Secret 生命周期策略定义见表 A.26，TCM 状态标记见表 A.27。

表 A.26 Secret 生命周期策略定义

属性	描述
TSM_SECRET_LIFETIME_ALWAYS	授权数据将一直有效

表 A.27 TCM 状态标记

Flag	描述	使用
TSM_TCMSTATUS_DISABLEOWNERCLEAR	永久禁止 TCM 所有者进行 ClearOwner 操作 此时，方法 ClearOwner() 中的 fForcedClear 参数将不再允许取 FALSE 值 这个设置需要所有者授权	SetStatus GetStatus
TSM_TCMSTATUS_DISABLEFORCECLEAR	临时禁止 TCM 所有者的强制清除操作（这种禁止只在本次系统运行时有效，在下一次系统重新启动时将被取消） 此时，方法 ClearOwner() 中的 fForcedClear 参数将暂时不允许取 TRUE 值（直到下次系统重新启动为止）	SetStatus GetStatus
TSM_TCMSTATUS_OWNERSETDISABLE	fTCMState = TRUE: 表示设置 TCM 的状态为 Disabled 时，不需要 TCM 所有者的授权	SetStatus GetStatus
TSM_TCMSTATUS_PHYSICALDISA BLE	fTCMState = TRUE: 表示设置 TCM 的状态为 Disabled 时必须是物理现场的	SetStatus GetStatus
TSM_TCMSTATUS_PHYSICALSETDEACTIVATED	fTCMState = TRUE: 表示设置 TCM 的状态为 Deactived 时必须是物理现场的	SetStatus GetStatus

Flag	描述	使用
TSM_TCMSTATUS_SETTEMPDEACTIVATED	暂时将TCM的状态设置为Deactived(直到下次系统重新启动为止)	SetStatus GetStatus
TSM_TCMSTATUS_SETOWNERINSTALL	fTCMState = TRUE: 表示允许使用TakeOwnership()方法来取得 TCM 的所有者关系 该操作需要物理现场	SetStatus GetStatus
TSM_TCMSTATUS_DISABLEPUBEKREAD	永久禁止在没有TCM所有者授权的情况下读取 EK 公钥信息的操作，即设置该属性后，必须有TCM所有者授权才能进行读取EK公钥信息的操作。 设置了这个属性后，GetPubEndorsementKey() 方法中的fOwnerAuthorized参数取FALSE值不可能再有效了 设置这个属性值需要所有者授权	SetStatus GetStatus
TSM_TCMSTATUS_DISABLED	将 TCM 设置为可用或不可用状态	SetStatus GetStatus
TSM_TCMSTATUS_DEACTIVATED	将 TCM 设置为激活或非激活状态	SetStatus GetStatus
TSM_TCMSTATUS_PHYSPRES_LIFETIMELOCK	*pfTCMState = TRUE: 表示在TCM生存期内，TCM的physicalPresenceHWEnable 和physicalPresenceCMDEnable 标志都不允许修改	GetStatus
TSM_TCMSTATUS_PHYSPRES_HWENABLE	*pfTCMState = TRUE: 表示TCM物理在线的硬件信号被允许可以用做物理在线的标志	GetStatus
TSM_TCMSTATUS_PHYSPRES_CMDENABLE	*pfTCMState = TRUE: 表示允许使用TCM命令TSC_PhysicalPresence来表明物理在线	GetStatus
TSM_TCMSTATUS_CEKP_USED	*pfTCMState = TRUE: 表明EK密钥对是使用CreateEndorsementKey()方法来生成 *pfTCMState = FALSE: 表明EK密钥对是由厂家创建	GetStatus
TSM_TCMSTATUS_PHYSPRESENCE	*pfTCMState = TRUE: 表示物理在线的软件标志	GetStatus
TSM_TCMSTATUS_PHYSPRES_LOCK	*pfTCMState = TRUE: 表示改变物理在线的标志的操作是不允许的	GetStatus

Flag	描述	使用
TSM_TCMSTATUS_ENABLE_REVOKEEK	表明是否允许EK被重新设置	GetStats
TSM_TCMSTATUS_NV_LOCK	*pfTCMState = TRUE: 表示NV授权访问是必需的	GetStats

A.1.2.13 算法标识符定义

算法标识符的定义基于 TSM_ALGORITHM_ID 数据类型。本规范涉及到的算法标识见表 A.28。

表 A.28 算法标识定义

算法标识符	描述
TCM_ALG_SM2	SM2 算法
TCM_ALG_SM4	SM4 密码算法
TCM_ALG_SCH	SM3 密码算法
TCM_ALG_HMAC	HMAC 算法
TCM_ALG_XOR	XOR 算法
TCM_ALG_KDF	KDF 算法

A.1.2.14 功能特性标记定义

以下标记定义了需要查询的功能特性，其定义是基于 TSM_FLAG 的。TCM 功能特性定义见表 A.29，永久性标记定义见 A.30，易失性标记见表 A.31，属性标记见表 A.32，TSM 功能特性见表 A.33。

表 A.29 TCM 功能特性定义

功能特性	描述
TSM_TCMCAP_ORD	查询TCM是否支持该命令
TSM_TCMCAP_ALG	查询是否支持该算法
TSM_TCMCAP_FLAG	返回所有永久性和易失性比特标志位的序列
TSM_TCMCAP_PROPERTY	判断 TCM 的物理特性
TSM_TCMCAP_VERSION	查询当前 TCM 版本。
TSM_TCMCAP_NV_LIST	获取用来定义 NV 存储区的索引列表
TSM_TCMCAP_NV_INDEX	获取指定的 NV 存储区的值
TSM_TCMCAP_MFR	获取厂商特定的TCM和TCM状态信息
TSM_TCMCAP_SYM_MODE	布尔值。查询 TCM 是否支持特定类型的对称加密
TSM_TCMCAP_HANDLE	返回 TCM 中当前已加载对象的句柄列表
TSM_TCMCAP_TRANS_ES	查询 TCM 是否在传输会话中支持特定的加密方案
TSM_TCMCAP_AUTH_ENCRYPT	查询TCM是否在授权会话的AuthData加密中支持特定的加密方案
TSM_TCSCAP_TRANSPORT	查询是否支持传输功能

Tspi_TCM_GetCapability()函数通过将 Capability 数据字段设置为 TSM_TCMCAP_FLAG，得到代表 TCM 功能特性的一串字节串，这些字符串是主机字节序（big-endian）的。

返回的前四字节表示UINT32形式的永久性标记,接下来的四个字节表示UINT32形式的易失性标记。永久性标记的 31-28 位在第一个字节返回，易失性标记的 3-0 比特在最后一个字节返回。位 0 定义为

UINT32 的第 1 位。

表 A.30　永久性标记

标记	值
TCM_PF_DISABLE	0x00000001
TCM_PF_OWNERSHIP	0x00000002
TCM_PF_DEACTIVATED	0x00000004
TCM_PF_READPUBEK	0x00000008
TCM_PF_DISABLEOWNERCLEAR	0x00000010
TCM_PF_PHYSICALPRESENCELIFETIMELOCK	0x00000040
TCM_PF_PHYSICALPRESENCEHWENABLE	0x00000080
TCM_PF_PHYSICALPRESENCECMDENABLE	0x00000100
TCM_PF_CEKPUSED	0x00000200
TCM_PF_TCMPOST	0x00000400
TCM_PF_TCMPOSTLOCK	0x00000800

表 A.31　易失性标记

标记	值
TCM_SF_DEACTIVATED	0x00000001
TCM_SF_DISABLEFORCECLEAR	0x00000002
TCM_SF_PHYSICALPRESENCE	0x00000004
TCM_SF_PHYSICALPRESENCELOCK	0x00000008

说明：没有在以上表中定义的比特位未被使用。

表 A.32　属性标记

功能特性	描述
TSM_TCSCAP_ALG	查询一个算法是否被支持
TSM_TCSCAP_VERSION	查询当前的 TSM 版本
TSM_TCSCAP_MANUFACTURER	查询当前的 TSM 厂商信息
TSM_TCSCAP_CACHING	查询是否支持密钥和授权的缓存
TSM_TCSCAP_PERSSTORAGE	查询是否支持永久性存储

表 A.33　TSM 功能特性

Capability Area	描述
TSM_TSPCAP_ALG	查询是否支持某个算法
TSM_TSPCAP_VERSION	查询当前的 TSM 版本
TSM_TSPCAP_PERSSTORAGE	查询是否支持永久性存储
TSM_TSPCAP_MANUFACTURER	查询当前的 TSM 厂商信息

A.1.2.15 子属性标记定义

子属性标记从属于属性标记，其定义基于 TSM_TCMCAP_PROPERTY。TCM 子属性标记见表 A.34，TSM 子属性见表 A.35，TSM_TCSCAP_CACHING 子属性定义见表 A.36，TSM 子属性见表 A.37。

表 A.34 TCM 子属性标记

子属性	返回值
TSM_TCMCAP_PROP_PCR	UINT32 值 返回 TCM 支持的 PCR 寄存器个数
TSM_TCMCAP_PROP_PCRMAP	返回 TCM_PCR_ATTRIBUTES 的比特标志位 TCM_PCR_ATTRIBUTES 是与 TSM_TCMCAPPROP_PCR 相关的
TSM_TCMCAP_PROP_MANUFACTURER	UINT32 值 返回 TCM 厂商的标识
TSM_TCMCAP_PROP_SLOTS 或 TSM_TCMCAP_PROP_KEYS	UINT32 值 返回 TCM 可以加载的 256 位 SM2 密钥的最大个数 可随时间和情况而改变
TSM_TCMCAP_PROP_OWNER	布尔值 返回 TRUE 表示 TCM 成功地创建了一个所有者
TSM_TCMCAP_PROP_MAXKEYS	UINT32 值 返回 TCM 所支持的 256 位 SM2 密钥的最大个数，不含 EK 和 SMK
TSM_TCMCAP_PROP_AUTHSESSIONS	UINT32 值 可用的授权会话的个数，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXAUTHSESSIONS	UINT32 值 返回 TCM 支持的可加载授权会话的最大个数，可随时间和情况而改变
TSM_TCMCAP_PROP_TRANSESSIONS	UINT32 值 返回可用传输会话的个数，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXTRANSESSIONS	UINT32 值 返回 TCM 支持的可加载传输会话的最大个数
TSM_TCMCAP_PROP_SESSIONS	UINT32 值 返回会话池中可用会话的个数 会话池中的会话包括授权会话和传输会话，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXSESSIONS	UINT32 值 返回 TCM 支持的最大会话个数，包括授权会话和传输会话
TSM_TCMCAP_PROP_CONTEXTS	UINT32 值 返回可保存的会话个数，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXCONTEXTS	UINT32 值 返回保存的会话的最大个数

子属性	返回值
TSM_TCMCAP_PROP_COUNTERS	UINT32 值 返回可用单调计数器的个数，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXCOUNTERS	UINT32 值 返回 TCM_CreateCounter 控制的单调计数器的最大个数
TSM_TCMCAP_PROP_MINCOUNTERINCTIME	UINT32 值 表示单调计数器递增的最小时间间隔，该间隔以 1/10 秒为度量单位
TSM_TCMCAP_PROP_ACTIVECOUNTER	返回当前计数器的 ID 若没有活动的计数器，则返回 0xff..ff
TSM_TCMCAP_PROP_TISTIMEOUTS	UINT32 值四元数组,每个元素表示以毫秒计的超时值，顺序如下：TIMEOUT_A, TIMEOUT_B, TIMEOUT_C, TIMEOUT_D 由平台特定的接口规范决定在何处使用这些超时值
TSM_TCMCAP_PROP_STARTUPEFFECTS	返回 TCM_STARTUP_EFFECTS 结构
TSM_TCMCAP_PROP_MAXCONTEXTCOUNTDIST	UINT32 值 返回上下文计数值的最大间距，至少必须为 2^16-1
TSM_TCMCAP_PROP_DURATION	返回 UINT32 值三元数组，每个元素依次表示如下三类命令以毫秒计的周期值： SMALL_DURATION,MEDIUM_DURATION, LONG_DURATION
TSM_TCMCAP_PROP_MAXNVAVAILABLE	UINT32 值 返回可分配的 NV 区域的最大个数，可随时间和情况而改变
TSM_TCMCAP_PROP_MAXNVWRITE	在 TCM 还没有所有者时，此值表示对 NV 进行写操作的次数
TSM_TCMCAP_PROP_REVISION	BYTE 用标准结构标识 TCM 的主版本号和次版本号，主版本号在前，次版本号在后
TSM_TCMCAP_PROP_LOCALITIES_AVAIL	TCM 中可用的 localities 的数目
TSM_TCMCAP_PROP_INPUTBUFFERSIZE	UINT32 返回 TCM 中用作输入的缓冲区的大小，以字节为单位

表 A.35 TSM 子属性

子属性	描述
TSM_TCSCAP_PROP_MANUFACTURER_STR	返回由 TSM 生产商指定内容的字符串，它的数据类型为 TSM_UNICODE
TSM_TCSCAP_PROP_MANUFACTURER_ID	返回一个 UINT32 类型的数值，标识特定的 TSM 生产商

表 A.36 TSM_TCSCAP_CACHING 子属性定义

子属性	描述
TSM_TCSCAP_PROP_KEYCACHE	类型为 TSM_BOOL 的数值，标识是否支持密钥缓存

子属性	描述
TSM_TCSCAP_PROP_AUTHCACHE	类型为 TSM_BOOL 的数值，标识是否支持授权会话缓存

表 A.37　TSM 子属性

子属性	描述
TSM_TSPCAP_PROP_MANUFACTURER_STR	返回由生产商定制内容的字符串，它的数据类型为 TSM_UNICODE
TSM_TSPCAP_PROP_MANUFACTURER_ID	返回一个UINT32类型的数值，标识特定的TSM生产商

A.1.2.16　永久存储区标记定义

永久存储区标记定义见表 A.38

表 A.38　永久存储区标记

永久性存储类型	描述
TSM_PS_TYPE_USER	密钥被注册到用户永久存储空间
TSM_PS_TYPE_SYSTEM	密钥被注册到系统永久存储空间

A.1.2.17　迁移方案定义

迁移方案定义见表 A.39。

表 A.39　迁移方案定义

迁移方案	描述
TSM_MS_MIGRATE	迁移时用来保护另一个密钥的密钥，用法是先调用函数 Tspi_Key_CreateMigrationBlob，再调用函数 Tspi_Key_ConvertMigrationBlob
TSM_MS_REWRAP	迁移保护密钥的公钥部分，通过调用函数 Tspi_Key_CreateMigrationBlob 对被迁移密钥进行重新加密。

A.1.2.18　密钥用途定义

密钥用途定义见表 A.40。

表 A.40　密钥用途定义

密钥用途	描述
TCM_SM2KEY_STORAGE	用于存储保护密钥树中其它密钥的 SM2 密钥
TCM_SM2KEY_SIGNING	用于签名的 SM2 密钥
TCM_SM2KEY_BIND	用于数据加解密的 SM2 密钥
TCM_SM2KEY_IDENTITY	SM2 密钥，该密钥只用于那些需要 TCM 身份的操作

密钥用途	描述
TCM_SM2KEY_MIGRATE	可用于密钥迁移的 SM2 密钥
TCM_SM2KEY_PEK	SM2 平台加密密钥
TCM_SM4KEY_STORAGE	用于存储保护密钥树中其它密钥的 SM4 密钥
TCM_SM4KEY_BIND	用于数据加解密的 SM4 密钥

A.1.2.19 密钥长度定义

密钥长度定义见表 A.41。

表 A.41 密钥长度定义

密钥长度	描述
TSM_KEY_SIZEVAL_128BIT	密钥长度为 128 位
TSM_KEY_SIZEVAL_256BIT	密钥长度为 256 位 (SM2 私钥)
TSM_KEY_SIZEVAL_512BIT	密钥长度为 512 位 (SM2 公钥)

A.1.2.20 密钥类型标记

密钥类型标记见表 A.42。

表 A.42 密钥类型标记定义

密钥类型	描述
TSM_KEYFLAG_MIGRATABLE	若密钥是可迁移的，则此值需设置为 1
TSM_KEYFLAG_VOLATILEKEY	若密钥是易失性的，则此值需设置为 1

A.1.2.21 密钥结构类型

下表定义的密钥数据结构由 Tspi_GetAttribUint32(TSM_TSPATTRIB_KEY_INFO,TSM_TSPATTRIB_KEYINFO_KEYSTRUCT) 返回，或者由 Tspi_SetAttribUint32 (TSM_TSPATTRIB_KEY_INFO,TSM_TSPATTRIB_KEYINFO_KEYSTRUCT) 设置。密钥结构类型定义见表 A.43。

表 A.43 密钥结构类型定义

密钥结构类型	描述
TSM_KEY_STRUCT_KEY	密钥对象采用 TCM 密钥结构

A.1.2.22 密钥授权

密钥授权定义见表 A.44。

表 A.44 密钥授权定义

密钥授权	描述
TSM_KEYAUTH_AUTH_NEVER	密钥不需要授权
TSM_KEYAUTH_AUTH_ALWAYS	密钥需要授权
TSM_KEYAUTH_AUTH_PRIV_USE_ONLY	此值表示如果有命令需要用到密钥的私钥部分，则一定需要授权，如果有命令只需使用密钥的公钥部分而不需要密

密钥授权	描述
	钥的私钥部分，则不需要授权

A.1.2.23 密钥使用方案定义

密钥使用方案定义见表 A.45。

表 A.45 密钥使用方案定义

算法标识符	描述
TSM_ES_NONE	没有设置加密方案
TSM_ES_SM2	采用 SM2 算法进行加密
TSM_ES_SM4_CBC	采用 SM4 CBC 方案进行加密
TSM_ES_SM4_ECB	采用 SM4 ECB 方案进行加密

A.1.2.24 签名方案定义

签名方案定义见表 A.46。

表 A.46 签名方案定义

算法标识	描述
TSM_SS_NONE	没有设置签名方案
TSM_SS_SM2	采用国标签名方案

A.1.2.25 PCR 结构类型

PCR 结构类型定义见表 A.47。

表 A.47 PCR 结构类型

PCR 结构类型	描述
TSM_PCRS_STRUCT_INFO	Pcr 对象采用 TCM 的 PCR 数据结构

A.1.2.26 共享秘密

共享秘密定义见表 A.48。

表 A.48 共享秘密

公开的共享秘密	描述
TSM_WELL_KNOWN_SECRET	32 位全零

A.2 数据结构

A.2.1 TSM_VERSION

定义:

```
typedef struct tdTSM_VERSION
{
        BYTE    bMajor;
        BYTE    bMinor;
```

```
        BYTE    bRevMajor;
        BYTE    bRevMinor;
    } TSM_VERSION;
```

成员变量:

——bMajor　　本 TSM 规范实现的主版本号标识码

——bMinor　　本 TSM 规范实现的次版本号标识码

——bRevMajor　TSM 厂商实现的主版本号取值，由 TSM 厂商确定

——bRevMinor　TSM 厂商实现的次版本号取值，由 TSM 厂商确定

A.2.2 TSM_PCR_EVENT

该结构提供有关单个 PCR 扩展事件的信息。

定义:

```
typedef struct tdTSM_PCR_EVENT
{
    TSM_VERSION     versionInfo;
    UINT32          ulPcrIndex;
    TSM_EVENTTYPE   eventType;
    UINT32          ulPcrValueLength;
    BYTE*           rgbPcrValue;
    UINT32          ulEventLength;
    BYTE*           rgbEvent;
} TSM_PCR_EVENT;
```

成员变量:

——versionInfo　　由 TSM 设置的版本数据

——ulPcrIndex　　由 TSM 设置的该事件所属的 PCR 索引

——eventType　　事件类型标记

——ulPcrValueLength　参数 rgbPcrValue 的数据长度(以字节为单位)，由 TSM 设置

——rgbPcrValue　　指向通过调用函数 Tspi_TCM_PcrExtend 扩展到 TCM 的值

——ulEventLength　　参数 rgbEvent 的数据长度(以字节为单位)

——rgbEvent　　PCR 事件信息

A.2.3 TSM_EVENT_CERT

证书结构，用于 TSM_EV_CODE_CERT 类型的事件

定义:

```
typedef struct tdTSM_EVENT_CERT
{
    TSM_VERSION         versionInfo;
    UINT32              ulCertificateHashLength
    BYTE*               rgbCertificateHash;
    UINT32              ulEntityDigestLength
    BYTE*               rgbEntityDigest;
    TSM_BOOL            fDigestChecked;
    TSM_BOOL            fDigestVerified;
```

```
    UINT32              ulIssuerLength;
    BYTE*               rgbIssuer;
} TSM_EVENT_CERT;
```

成员变量:

——versionInfo	版本信息
——ulCertificateHashLength	参数 rgbCertificatHash 的数据长度（以字节为单位）
——rgbCertificateHash	指向整个 VE(Validation Entity)证书的杂凑值
——ulEntityDigestLength	参数 rgbEntityDigest 的数据长度（以字节为单位）
——rgbEntityDigest	指向整个证书的实际摘要值
——fDigestChecked	如果此值为 TRUE，则需要将整个事件日志与证书中的摘要值进行比较；如果此值为 FALSE，则不需要检查
——fDigestVerified	只有在 fDigestChecked 为 TRUE 的情况下，此值才起作用。如果度量值与证书中的摘要值匹配，则此值为 TRUE，否则为 FALSE
——ulIssuerLength	参数 rgbIssuer 的数据长度（以字节为单位）
——rgbIssuer	指向颁发者证书

A.2.4 TSM_UUID

本数据结构提供关于一个 UUID 标识符的信息，该标识符在一个给定平台密钥树范围内是唯一的。有些 UUID 专门为一些特定密钥保留，比如 SMK。

这些 UUID 用于在 TSM 密钥管理器的永久性存储区中注册密钥。

UUID 定义遵从 IEEE 802。

定义:

```
typedef struct tdTSM_UUID
{
    UINT32      ulTimeLow;
    UINT16      usTimeMid;
    UINT16      usTimeHigh;
    BYTE        bClockSeqHigh;
    BYTE        bClockSeqLow;
    BYTE        rgbNode[6];
} TSM_UUID;
```

成员变量:

——ulTimeLow	时间戳的低字段
——usTimeMid	时间戳的中间字段
——usTimeHigh	时间戳的高字段乘以版本号
——bClockSeqHigh	时钟序列高字段乘以变量
——bClockSeqLow	时钟序列低字段
——rgbNode	唯一性节点标识符

A.2.5 TSM_KM_KEYINFO

TSM_KM_KEYINFO 数据结构提供关于注册在 TSM 永久性存储区的密钥信息。

定义:

```
typedef struct tdTSM_KM_KEYINFO
{
    TSM_VERSION         versionInfo;
    TSM_UUID            keyUUID;
    TSM_UUID            parentKeyUUID;
    BYTE                bAuthDataUsage;
    TSM_FLAG            persistentStorageType;
    TSM_FLAG            persistentStorageTypeParent;
    TSM_BOOL            fIsLoaded; // TRUE: actually loaded in TCM
    UINT32              ulVendorDataLength;// may be 0
    BYTE*               rgbVendorData; // may be NULL
} TSM_KM_KEYINFO;
```

成员变量:

——versionInfo　　版本数据

——keyUUID　　注册在 TSM 密钥管理器永久性存储区中的密钥 UUID

——parentKeyUUID　　注册在 TSM 密钥管理器永久性存储区中的父密钥 UUID，用来加密由 keyUUID 寻址的密钥

——bAuthDataUsage　　表示使用密钥是否需要授权的标记，目前定义的值为 0x00 与 0x01，值 0x00 表示密钥使用不需要授权，值 0x01 表示每次密钥使用都需要进行授权，其它值将保留给未来版本使用。

——persistentStorageType　　标记，表示密钥注册的永久性存储区

——persistentStorageTypeParent 标记，表示父密钥注册的永久性存储区

——fIsLoaded　　标记，表示该密钥是否加载到 TCM 中
TRUE 表示密　钥已加载到 TCM 中，FALSE 表示密钥未加载到 TCM 中

——ulVendorDataLength　　参数 rgbVendorData 的长度(字节)

——rgbVendorData：　　指向厂商特定数据的指针

A.2.6 TSM_VALIDATION

TSM_VALIDATION 数据结构用于验证数字签名。

下列函数使用该数据结构：

Tspi_TCM_CertifySelfTest,
Tspi_TCM_GetCapabilitySigned,
Tspi_Key_CertifyKey,
Tspi_TCM_CreateEndorsementKey,
Tspi_TCM_GetPubEndorsementKey
Tspi_TCM_CreateRevocableEndorsementKey
Tspi_TCM_Quote

Tspi_Context_CloseSignTransport

定义:

```
typedef struct tdTSM_VALIDATION
{
    TSM_VERSION     versionInfo;
    UINT32          ulExternalDataLength;
    BYTE*           rgbExternalData;
    UINT32          ulDataLength;
    BYTE*           rgbData;
    UINT32          ulValidationLength;
    BYTE*           rgbValdationData;
} TSM_VALIDATION;
```

成员变量:

——versionInfo 版本数据
——ulExternalData rgbExternalData 的长度(字节)
——rgbExternalData 内存指针，该内存包含用于抗重放攻击的随机数
——ulDataLength rgbData 的长度(字节)
——rgbData 用于计算验证有效性的数据
——ulValidationLength rgbValidationData 的长度(字节)
——rgbValidationData 指向验证数据的指针

A.2.6.1 TCM_COUNTER_VALUE

本数据结构返回计数器值。计数器长度为 4 个字节。

定义:

```
typedef struct tdTCM_COUNTER_VALUE
{
    TCM_STRUCTURE_TAG   tag;
    BYTE                label[4];
    TCM_ACTUAL_COUNT    counter;
} TCM_COUNTER_VALUE;
```

成员变量:

——tag 标记，对计数器，该值为 0x000E
——label 计数器标签(4 字节)
——counter 32 位的计数器值

A.2.6.2 TSM_ENVELOP

本数据结构用于数字信封函数

定义:

```
typedef struct tdTSM_ENVELOP
{
    UINT32      encSymSize;
    BYTE*       encSym; // 对 TCM_STORE_SYMKEY 结构加密后的数据缓冲区
    UINT32      dataSize;
```

```
    BYTE*       data;
}TSM_ENVELOP;
```

成员变量:

——encSymSize　encSym 的数据长度(字节)

——encSym　　被加密的对称密钥

——dataSize　　参数 data 的长度

——data　　用对称密钥加密的数据

A.3 授权数据处理

TSM 提供策略对象帮助调用程序处理和缓存授权对象的秘密。

下列对象是授权对象：

a) TCM；

b) 密钥；

c) 加密数据。

TSM 知道何时使用对象的授权信息或者从对象授权信息衍生出来的会话授权信息。一个授权对象只指派一个策略对象，但一个策略对象可以指派给 0 到 n 个授权对象，以方便多个授权对象拥有相同的授权信息。如果一个授权对象没有明确地分配一个策略对象，TSM 自动给它分配默认的策略对象。

当调用需要授权的函数时，应用程序不需要提供授权数据。这个服务不是限制性的，可通过应用程序根据策略对象调整。服务提供者应该使用非分页内存存放提供给它的秘密，这是依赖平台的。释放此内存前，首先将内存区清零。

策略对象的默认模式是 TSM_SECRET_MODE_PLAIN, 也可设置为 TSM_SECRET_MODE_POPUP, TSM_SECRET_MODE_SCH

模式：

TSM_SECRET_MODE_POPUP

显示一个输入密码的对话框。用户提供的密码被当作一个以 null 结尾的字符串来处理，并使用 SM3 作杂凑处理得到授权信息。一旦提供了此信息，可能在适当的策略对象中缓存授权信息（依赖策略对象的设置），这样对话框不会再次弹出.

TSM_SECRET_MODE_PLAIN 或 TSM_SECRET_MODE_SM3

应用程序可以基于每个策略设置授权信息。

此授权信息可为需要授权的命令提供授权。

策略对象模式定义见表 A.49。

表 A.49　策略对象模式

类型	定义
TSM_SECRET_MODE_SCH	输入经外部杂凑计算得到的 32 个字节数组作为授权数据
TSM_SECRET_MODE_PLAIN	输入一个任意字节的数组，然后将其进行杂凑运算，作为授权数据
TSM_SECRET_MODE_NONE	对于工作对象没有授权数据。

A.4 返回码定义

TCM 服务模块返回码定义见表 A.50，TCS 返回吗定义见表 A.51

表 A.50　TCM 服务模块返回码

定义类型	预定义值	说明
TSM_E_BASE	0x00000000L	
TSM_SUCCESS	0x00000000L	命令处理成功
TSM_E_FAIL	TSM_E_BASE + 0x002L	检测到内部错误
TSM_E_BAD_PARAMETER	TSM_E_BASE + 0x003L	一个或多个参数错误
TSM_E_INTERNAL_ERROR	TSM_E_BASE + 0x004L	TCM 内部错误
TSM_E_OUTOFMEMORY	TSM_E_BASE + 0x005L	超出内存
TSM_E_NOTIMPL	TSM_E_BASE + 0x006L	未执行
TSM_E_KEY_ALREADY_REGISTERED	TSM_E_BASE + 0x008L	密钥已经注册
TSM_E_TCM_UNEXPECTED	TSM_E_BASE + 0x010L	非期望的 TCM 错误
TSM_E_COMM_FAILURE	TSM_E_BASE + 0x011L	通信错误
TSM_E_TIMEOUT	TSM_E_BASE + 0x012L	操作超时
TSM_E_TCM_UNSUPPORTED_FEATURE	TSM_E_BASE + 0x014L	不支持的请求
TSM_E_CANCELED	TSM_E_BASE + 0x016L	操作被取消
TSM_E_PS_KEY_NOTFOUND	TSM_E_BASE + 0x020L	密钥不在永久存储区域
TSM_E_PS_KEY_EXISTS	TSM_E_BASE + 0x021L	密钥已经在永久存储区域存在
TSM_E_PS_BAD_KEY_STATE	TSM_E_BASE + 0x022L	在永久存储区域设置密钥数据无效
TSM_E_INVALID_OBJECT_TYPE	TSM_E_BASE + 0x101L	无效的操作对象类型
TSM_E_NO_CONNECTION	TSM_E_BASE + 0x102L	核心服务连接不存在
TSM_E_CONNECTION_FAILED	TSM_E_BASE + 0x103L	核心服务连接失败
TSM_E_CONNECTION_BROKEN	TSM_E_BASE + 0x104L	与核心服务通信失败
TSM_E_HASH_INVALID_ALG	TSM_E_BASE + 0x105L	无效的密码杂凑算法
TSM_E_HASH_INVALID_LENGTH	TSM_E_BASE + 0x106L	杂凑长度与算法不符
TSM_E_HASH_NO_DATA	TSM_E_BASE + 0x107L	杂凑对象没有数据
TSM_E_INVALID_ATTRIB_FLAG	TSM_E_BASE + 0x109L	属性标记设置错误
TSM_E_INVALID_ATTRIB_SUBFLAG	TSM_E_BASE + 0x10AL	子属性标记设置错误
TSM_E_INVALID_ATTRIB_DATA	TSM_E_BASE + 0x10BL	数据属性错误
TSM_E_INVALID_OBJECT_INIT_FLAG	TSM_E_BASE + 0x10CL	创建错误对象的标志信息
TSM_E_NO_PCRS_SET	TSM_E_BASE + 0x10DL	没有选择或设置 PCR 寄存器
TSM_E_KEY_NOT_LOADED	TSM_E_BASE + 0x10EL	指定密钥没有被加载
TSM_E_KEY_NOT_SET	TSM_E_BASE + 0x10FL	密钥信息不可用
TSM_E_VALIDATION_FAILED	TSM_E_BASE + 0x110L	数据有效性验证失败
TSM_E_TSP_AUTHREQUIRED	TSM_E_BASE + 0x111L	需要授权
TSM_E_TSP_AUTH2REQUIRED	TSM_E_BASE + 0x112L	需要多授权
TSM_E_TSP_AUTHFAIL	TSM_E_BASE + 0x113L	授权错误
TSM_E_TSP_AUTH2FAIL	TSM_E_BASE + 0x114L	多授权错误
TSM_E_KEY_NO_MIGRATION_POLICY	TSM_E_BASE + 0x115L	没有为密钥设置迁移策略
TSM_E_POLICY_NO_SECRET	TSM_E_BASE + 0x116L	指定的策略对象没有设置授权信息
TSM_E_INVALID_OBJ_ACCESS	TSM_E_BASE + 0x117L	对象状态错误导致操作失败
TSM_E_INVALID_ENCSCHEME	TSM_E_BASE + 0x118L	无效的加密方案标识
TSM_E_INVALID_SIGSCHEME	TSM_E_BASE + 0x119L	无效的签名方案标识
TSM_E_ENC_INVALID_LENGTH	TSM_E_BASE + 0x120L	无效的加密数据长度

定义类型	预定义值	说明
TSM_E_ENC_NO_DATA	TSM_E_BASE + 0x121L	无数据供加密
TSM_E_ENC_INVALID_TYPE	TSM_E_BASE + 0x122L	无效的加密类型
TSM_E_INVALID_KEYUSAGE	TSM_E_BASE + 0x123L	无效的密钥类型
TSM_E_VERIFICATION_FAILED	TSM_E_BASE + 0x124L	签名验证错误
TSM_E_HASH_NO_IDENTIFIER	TSM_E_BASE + 0x125L	未定义的密码杂凑算法标识
TSM_E_INVALID_HANDLE	TSM_E_BASE + 0x126L	无效的句柄
TSM_E_SILENT_CONTEXT	TSM_E_BASE + 0x127L	需要用户输入静默的上下文
TSM_E_EK_CHECKSUM	TSM_E_BASE + 0x128L	TSP 验证 EK 校验和失败
TSM_E_INVALID_RESOURCE	TSM_E_BASE + 0x13AL	内存指针错误
TSM_E_NV_AREA_EXIST	TSM_E_BASE + 0x13BL	对已经使用的 NV 区重新定义
TSM_E_NV_AREA_NOT_EXIST	TSM_E_BASE + 0x13CL	定义的 NV 区域不存在
TSM_E_TSP_TRANS_AUTHFAIL	TSM_E_BASE + 0x13DL	传输会话授权失败
TSM_E_TSP_TRANS_AUTHREQUIRED	TSM_E_BASE + 0x13EL	传输会话需要授权
TSM_E_TSP_TRANS_NOTEXCLUSIVE	TSM_E_BASE + 0x13FL	不在传输会话范围内
TSM_E_TSP_TRANS_FAIL	TSM_E_BASE + 0x140L	产生了一个传输保护的错误
TSM_E_TSP_ERROR_KEYTYPE	TSM_E_BASE + 0x141L	错误的密钥类型
TSM_E_TSP_APCREATE_FAIL	TSM_E_BASE + 0x142L	APCreate 授权失败错误
TSM_E_TSP_ENCRYPT_ERROR	TSM_E_BASE + 0x143L	加密错误
TSM_E_EXCHANGE_HANDLE_EXIST	TSM_E_BASE + 0x145L	密钥交换句柄已经存在
TSM_E_EXCHANGE_HANDLE_NOT_EXIST	TSM_E_BASE + 0x146L	密钥交换句柄不存在

表A.51　TCS的返回码

定义类型	预定义值	说明
TSM_E_BASE	0x00000000L	
TCS_SUCCESS	TSM_SUCCESS	命令处理成功
TCS_E_FAIL	TSM_E_BASE + 0x002L	其他错误
TCS_E_INVALID_CONTEXTHANDLE	TSM_E_BASE + 0x0C1L	无效的线上下文句柄
TCS_E_INVALID_KEYHANDLE	TSM_E_BASE + 0x0C2L	无效的密钥句柄
TCS_E_INVALID_AUTHHANDLE	TSM_E_BASE + 0x0C3L	无效的授权会话句柄
TCS_E_INVALID_AUTHSESSION	TSM_E_BASE + 0x0C4L	会话句柄已关闭
TCS_E_INVALID_KEY	TSM_E_BASE + 0x0C5L	密钥未加载
TCS_E_KEY_MISMATCH	TSM_E_BASE + 0x0C8L	密钥不匹配
TCS_E_KM_LOADFAILED	TSM_E_BASE + 0x0CAL	UUID 标识的密钥加载失败（其中的恶一个父密钥加载需要授权）
TCS_E_BAD_INDEX	TSM_E_BASE + 0x0CDL	错误的内存索引
TCS_E_KEY_ALREADY_REGISTERED	TSM_E_KEY_ALREADY_REGISTERED	参见 TSM 的返回码定义
TCS_E_BAD_PARAMETER	TSM_E_BAD_PARAMETER	
TCS_E_OUTOFMEMORY	TSM_E_OUTOFMEMORY	
TCS_E_NOTIMPL	TSM_E_NOTIMPL	

定义类型	预定义值	说明
TCS_E_INTERNAL_ERROR	TSM_E_INTERNAL_ERROR	