目录

[1. 可信软件栈基本功能接口 3](#_Toc484422672)

[1.1 System API 上下文管理 3](#_Toc484422673)

[1.1.1 软件栈上下文初始化 3](#_Toc484422674)

[1.1.2 软件栈上下文注销 3](#_Toc484422675)

[1.1.3 查询创建一个上下文结构体实例所需分配内存 4](#_Toc484422676)

[1.1.4 查询当前绑定的命令传输接口 4](#_Toc484422677)

[1.1.5 取回应答桢授权区域数据 5](#_Toc484422678)

[1.1.6 设置命令帧授权区域 5](#_Toc484422679)

[2. 标准 TPM 命令 6](#_Toc484422680)

[2.1 开机关机命令 6](#_Toc484422681)

[2.1.1 开机命令 6](#_Toc484422682)

[2.1.2 关机命令 6](#_Toc484422683)

[2.2 设备资源管理命令 7](#_Toc484422684)

[2.2.1 查询平台功能和限制 7](#_Toc484422685)

[2.3 密钥管理命令 8](#_Toc484422686)

[2.3.1 读取节点公开数据 8](#_Toc484422687)

[2.3.2 密钥对象的创建、密钥树节点的加载 10](#_Toc484422688)

[3. 其他命令 19](#_Toc484422689)

[3.1 其他命令 20](#_Toc484422690)

[3.1.1 Tss2\_Sys\_EvictControl 20](#_Toc484422691)

[4. PCR相关命令 21](#_Toc484422692)

[4.1 PCR相关命令 21](#_Toc484422693)

[4.1.1 PCR Allocate 21](#_Toc484422694)

[4.1.2 PCR Extend 22](#_Toc484422695)

[4.1.3 PCR Read 23](#_Toc484422696)

[4.1.4 PCR Event 24](#_Toc484422697)

[4.1.5 PCR SetAuthPolicy 25](#_Toc484422698)

[4.1.6 PCR SetAuthValue 26](#_Toc484422699)

[4.1.7 PCR Reset 27](#_Toc484422700)

[5. 随机数相关命令 27](#_Toc484422701)

[5.1 随机数相关命令 27](#_Toc484422702)

[5.1.1 GetRandom 27](#_Toc484422703)

[5.1.2 StirRandom 28](#_Toc484422704)

[6. Clock相关命令 29](#_Toc484422705)

[6.1 Clock相关命令 29](#_Toc484422706)

[6.1.1 Read clock 29](#_Toc484422707)

[6.1.2 ClockSet 29](#_Toc484422708)

[6.1.3 ClockRateAdjust 30](#_Toc484422709)

[7. NV相关命令 31](#_Toc484422710)

[7.1 NV相关命令 31](#_Toc484422711)

[7.1.1 Tss2\_Sys\_NV\_DefineSpace 31](#_Toc484422712)

[7.1.2 Tss2\_Sys\_NV\_UndefineSpace 35](#_Toc484422713)

[7.1.3 Tss2\_Sys\_NV\_UndefineSpaceSpecial 36](#_Toc484422714)

[7.1.4 Tss2\_Sys\_NV\_Read 36](#_Toc484422715)

[7.1.5 Tss2\_Sys\_NV\_ Write 37](#_Toc484422716)

[7.1.6 Tss2\_Sys\_NV\_ ReadLock 38](#_Toc484422717)

[7.1.7 Tss2\_Sys\_NV\_ WriteLock 38](#_Toc484422718)

[7.1.8 Tss2\_Sys\_NV\_ReadPublic 39](#_Toc484422719)

[7.1.9 Tss2\_Sys\_NV\_ GlobalWriteLock 39](#_Toc484422720)

[7.1.10 Tss2\_Sys\_NV\_ Increment 40](#_Toc484422721)

[7.1.11 Tss2\_Sys\_NV\_ Extend 40](#_Toc484422722)

[7.1.12 Tss2\_Sys\_NV\_ SetBits 41](#_Toc484422723)

[7.1.13 Tss2\_Sys\_NV\_ ChangeAuth（未验证） 41](#_Toc484422724)

[7.1.14 Tss2\_Sys\_NV\_ Certify（未验证） 42](#_Toc484422725)

[8. Session相关命令 42](#_Toc484422726)

[8.1 Session相关命令 42](#_Toc484422727)

[8.1.1 Tss2\_Sys\_StartAuthSession 42](#_Toc484422728)

[8.1.2 Tss2\_Sys\_GetSessionAuditDigest(未验证) 45](#_Toc484422729)

[9. Policy相关命令 45](#_Toc484422730)

[9.1 Policy相关命令 45](#_Toc484422731)

[9.1.1 Tss2\_Sys\_PolicyNV 45](#_Toc484422732)

[9.1.2 Tss2\_Sys\_PolicyNvWritten 47](#_Toc484422733)

1. 可信软件栈基本功能接口

本章介绍基本的API接口，包括软件栈上下文的初始化、命令桢/应答桢报文数据的封装与拆分。

* 1. System API 上下文管理

上下文管理类用于维护 TSS 软件栈自身执行环境的相关信息，其功能主要包括与 TPM 资源管理器软件模块之间的通信管理，以及软件栈自身的内存分配管理。

* + 1. 软件栈上下文初始化

**函数名：**Tss2\_Sys\_Initialize ()

**功能描述：**

软件栈上下文数据初始化。

**接口定义：**

TSS2\_RC Tss2\_Sys\_Initialize(

TSS2\_SYS\_CONTEXT\* pSysContext,// IN-OUT

size\_t contextSize,// IN

TSS2\_TCTI\_CONTEXT\* pTctiContext,// IN

TSS2\_ABI\_VERSION\* pAbiVersion // IN

);

**输入参数：**

pSysContext：输入指向已经预分配空间但尚未经过初始化的上下文结构体的指针。

contextSize：指明当前预先为上下文结构体 pSysContext 分配的内存空间大小。

pTctiContext：设置软件栈要使用的可信命令传输接口(TCTI)上下文结构体。

pAbiVersion：指明二进制应用程序编译环境所安装的 ABI 版本以及厂商编码。

**输出参数：**

pSysContext：输出经过数据初始化后的上下文结构体内容。

**返回值：**

TSS2\_RC\_SUCCESS：初始化成功。

TSS2\_SYS\_RC\_BAD\_CONTEXT：指向上下文结构体的指针无效。

TSS2\_SYS\_RC\_BAD\_TCTI\_STRUCTURE：命令传输接口上下文结构体指针无效。

TSS2\_SYS\_RC\_ABI\_MISMATCH：系统级 ABI 版本协商结果不匹配。

TSS2\_SYS\_RC\_INSUFFICIENT\_BUFFER：预先分配的内存缓冲区不足以完成初始化。

TSS2\_SYS\_RC\_GENERAL\_FAILURE：发生其他未知的内部错误导致初始化失败。

* + 1. 软件栈上下文注销

**功能描述：**

清空软件栈上下文结构体的所有内部数据，但不删除结构体本身占用的内存资源。

**接口定义：**

TSS2\_RC Tss2\_Sys\_Finalize(

TSS2\_SYS\_CONTEXT\* pSysContext// IN

);

**输入参数：**

pSysContext：软件栈上下文对象指针。

**输出参数：**

无。

**返回值：**

TSS2\_RC\_SUCCESS：上下文注销成功。

TSS2\_SYS\_RC\_BAD\_CONTEXT：指向上下文结构体的指针无效。

TSS2\_SYS\_RC\_GENERAL\_FAILURE：发生其他未知的内部错误导致注销失败。

* + 1. 查询创建一个上下文结构体实例所需分配内存

**功能描述：**

查询上下文结构体自身所需占用的内存空间的最小长度，以便调用者能够提前分配适当的内存空间预留给底层内部使用。

**接口定义：**

size\_t Tss2\_Sys\_GetContextSize(

size\_t maxCommandResponseSize // IN

);

**输入参数：**

maxCommandResponseSize：指定命令/应答桢报文缓冲区长度最大值，可以设置为 0 代表使用最大默认值。

**输出参数：**

无。

**返回值：**

无符号整数，表示上下文结构体最少需要占用的内存空间，正常情况下返回值等于 maxCommandResponseSize\*2 + 软件栈自身结构体成员占用的空间。

* + 1. 查询当前绑定的命令传输接口

**函数名：**Tss2\_Sys\_GetTctiContext()

**功能描述：**

查询当前上下文所绑定的命令传输接口(TCTI)结构体。

**接口定义：**

TSS2\_RC Tss2\_Sys\_GetTctiContext(

TSS2\_SYS\_CONTEXT\* pSysContext,// IN

TSS2\_TCTI\_CONTEXT\*\* ppTctiContext// OUT

);

**输入参数：**

pSysContext：上下文结构体指针。

**输出参数：**

ppTctiContext：更新此结构体指针的值，使其指向当前已绑定的 TCTI 命令传输接口结构体数据。

**返回值：**

TSS2\_RC\_SUCCESS：操作成功。

TSS2\_SYS\_RC\_BAD\_CONTEXT：指向上下文结构体的指针无效。

TSS2\_SYS\_RC\_GENERAL\_FAILURE：发生其他未知的内部错误导致操作失败。

* + 1. 取回应答桢授权区域数据

**函数名：**Tss2\_Sys\_GetRspAuths()

**功能描述：**

取回应答桢中的授权区域原始数据，便于后续计算应答桢 HMAC 校验值判断所传输数据的完整性。

**接口定义：**

TSS2\_RC Tss2\_Sys\_GetRspAuths(

TSS2\_SYS\_CONTEXT\* pSysContext, // IN

TSS2\_SYS\_RSP\_AUTHS\* rspAuthsArray // OUT

);

**输入参数：**

pSysContext：上下文结构体指针。

**输出参数：**

rspAuthsArray：输出授权区域详细数据。

**返回值：**

TSS2\_RC\_SUCCESS：操作成功。

TSS2\_SYS\_RC\_BAD\_CONTEXT：指向上下文结构体的指针无效。

TSS2\_SYS\_RC\_BAD\_SEQUENCE：调用次序错误。

TSS2\_SYS\_RC\_INVALID\_SESSIONS：授权区域数据块个数与命令桢提供的个数不一致。

TSS2\_SYS\_RC\_GENERAL\_FAILURE：发生其他未知的内部错误导致操作失败。

* + 1. 设置命令帧授权区域

**函数名：**Tss2\_Sys\_SetCmdAuths()

**功能描述：**

设置命令帧授权区域的内容。

**接口定义：**

TSS2\_RC Tss2\_Sys\_SetCmdAuths(

TSS2\_SYS\_CONTEXT\* pSysContext, // IN

TSS2\_SYS\_CMD\_AUTHS\* cmdAuthsArray // IN

);

**输入参数：**

pSysContext：上下文结构体指针。

cmdAuthsArray：命令帧授权区域的内容。

**输出参数：**

无。

**返回值：**

TSS2\_RC\_SUCCESS：操作成功。

TSS2\_SYS\_RC\_BAD\_CONTEXT：指向上下文结构体的指针无效。

TSS2\_SYS\_RC\_BAD\_SEQUENCE：调用次序错误。

TSS2\_SYS\_RC\_GENERAL\_FAILURE：发生其他未知的内部错误导致操作失败。

1. 标准 TPM 命令
   1. 开机关机命令
      1. 开机命令

**函数名：**Tss2\_Sys\_Startup()

**功能描述：**

发送开机命令，对 TPM 模块进行初始化。

**接口定义：**

TPM\_RC Tss2\_Sys\_Startup(

TSS2\_SYS\_CONTEXT\* pSysContext, // IN

TPM\_SUstartupType // IN

);

**输入参数：**

pSysContext：上下文结构体。

startupType：是否恢复上次关机时保存的运行状态，如下表：

|  |  |  |
| --- | --- | --- |
| **序号** | **开机选项** | **含义** |
| 1 | TPM\_SU\_CLEAR | 开机时清除全部临时数据 |
| 2 | TPM\_SU\_STATE | 开机时恢复上次关机前的运行状态 |

**输出参数：**

无。

**返回值：**

TPM\_RC\_SUCCESS：表示开机成功。

* + 1. 关机命令

**函数名：**Tss2\_Sys\_Shutdown ()

**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：**

TPM\_RC Tss2\_Sys\_Shutdown(

TSS2\_SYS\_CONTEXT\* pSysContext, // IN

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, // IN

TPM\_SUshutdownType, // IN

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray // OUT

);

**输入参数：**

pSysContext：上下文结构体指针。

cmdAuthsArray：命令帧授权区域的内容，备用参数向前兼容历史版本。

shutdownType：关机时是否保存当前的运行状态以供下次开机时可以恢复，如下表：

|  |  |  |
| --- | --- | --- |
| **序号** | **关机选项** | **含义** |
| 1 | TPM\_SU\_CLEAR | 关机时清除全部临时数据 |
| 2 | TPM\_SU\_STATE | 关机时保存当前的运行状态 |

**输出参数：**

rspAuthsArray：输出授权区域详细数据，备用参数向前兼容历史版本。

**返回值：**

TPM\_RC\_SUCCESS：表示关机成功。

* 1. 设备资源管理命令
     1. 查询平台功能和限制

**函数名：**Tss2\_Sys\_GetCapability ()

**功能描述：**

发送 TPM 2.0 命令桢：查询 TPM 模块的功能和内部属性限制。

**接口定义：**

TPM\_RC Tss2\_Sys\_GetCapability(

TSS2\_SYS\_CONTEXT \*pSysContext, // IN

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, // IN

TPM\_CAP capability, // IN

UINT32 property, // IN

UINT32 propertyCount, // IN

TPMI\_YES\_NO\* moreData, // OUT

TPMS\_CAPABILITY\_DATA\* capabilityData, // OUT

TSS2\_SYS\_RSP\_AUTHS\* rspAuthsArray // OUT

);

**输入参数：**

pSysContext：上下文结构体指针。

cmdAuthsArray：命令帧授权区域的内容。

capability：所要查询的属性类别，如下表：

|  |  |  |
| --- | --- | --- |
| **序号** | **类别 ID** | **含义** |
| 1 | TPM\_CAP\_ALGS | 查询TPM模块支持哪些密码算法 |
| 2 | TPM\_CAP\_HANDLES | 查询TPM模块支持哪些对象句柄 |
| 3 | TPM\_CAP\_COMMANDS | 查询TPM模块支持的命令编码集 |
| 4 | TPM\_CAP\_TPM\_PROPERTIES | 查询TPM模块最大容量参数 |

property：所要查询的具体子属性名称。当属性类别 capability=TPM\_CAP\_TPM\_PROPERTIES 时，子属性 property 的取值如下表：

|  |  |  |
| --- | --- | --- |
| **序号** | **子属性名称 ID** | **含义** |
| 1 | TPM\_PT\_MAX\_COMMAND\_SIZE | 查询命令桢最大总字节数 |
| 2 | TPM\_PT\_MAX\_RESPONSE\_SIZE | 查询应答桢最大总字节数 |
| 3 | TPM\_PT\_ACTIVE\_SESSIONS\_MAX | 查询最大支持的会话数量 |
| 4 | TPM\_PT\_HR\_LOADED | 查询当前已经加载的会话数量 |
| 5 | TPM\_PT\_CONTEXT\_GAP\_MAX | 查询最大可缓存密钥数量 |
| 6 | TPM\_PT\_TOTAL\_COMMANDS | 查询模块支持的全部命令总条数 |

propertyCount：本次查询其中多少条命令集。

**输出参数：**

moreData：输出一个标记位指示是否存在更多数据。

capabilityData：输出本次查询结果。

rspAuthsArray：输出应答桢授权区域详细数据。

**返回值：**

TPM\_RC\_SUCCESS：查询操作成功。

TPM\_RC\_INITIALIZE：TPM 模块未初始化，正在等待第一条开机命令

* 1. 密钥管理命令
     1. 读取节点公开数据

**函数名：**Tss2\_Sys\_ReadPublic()

**功能描述：**

读取密钥节点的公开数据。

**接口定义：**

TPM\_RC Tss2\_Sys\_ReadPublic(

TSS2\_SYS\_CONTEXT \*sysContext, // IN

TPMI\_RH\_HIERARCHY objectHandle, // IN

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, // IN

TPM2B\_PUBLIC \*outPublic,// OUT

TPM2B\_NAME \*name,// OUT

TPM2B\_NAME \*qualifiedName, // OUT

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray // OUT

);

**输入输出参数：**

| 上下文指针参数 | 数据结构 | 用途 |
| --- | --- | --- |
| sysContext | TSS2\_SYS\_CONTEXT | System API 上下文 |
| 对象句柄 | 数据结构 | 句柄取值范围 |
| objectHandle | TPMI\_RH\_HIERARCHY | 0x40开头的句柄是TPM内置句柄(Permanent)  0x80开头的句柄是临时句柄(Transient),重启后丢失  0x81开头的句柄重启后仍然有效(Persistent) |
| 授权信息 | 数据结构 | 用途 |
| cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS | 输入参数，填写命令桢授权会话区数据 |
| rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS | 输出参数，取回应答桢授权会话授权结果 |

| 应答桢  输出参数 | 数据结构 |  |
| --- | --- | --- |
| outPublic | 【TPM2B\_PUBLIC】 | **outPublic**  │  ├UINT16 size  └**publicArea**   │   ├**type**=密钥类型(例如：TPM\_ALG\_RSA)   │   ├**nameAlg**=指定哈希算法名(例如：TPM\_ALG\_SHA1)   │   ├**objectAttributes**=属性定义由32个标识位组成   │├a)密钥用途：签名/加解密/受限制   │├b)是否要求授权访问(userWithAuth, adminWithPolicy, noDA)   │├c)duplication (fixedParent, fixedTPM, encryptedDuplication)   │├d)sensitiveDataOrigin   │└e)密钥是否会被清除(stClear)   │   ├**authPolicy**=授权策略的校验值,当adminWithPolicy属性标识位被置1时有效   │├UINT16 size   │└BYTE buffer[]   │   ├**parameters**=密钥细节参数,具体解析格式分下列5种情况   ││1作为keyed hash参数   ││2作为对称密钥参数   ││3非对称密钥：作为RSA密钥的详细参数**rsaDetail**   ││4非对称密钥：作为椭圆曲线密钥的详细参数eccDetail   │└5作为非对称密钥通用参数   │   └**unique**=特征码,解析格式分下列4种情况    │1作为keyed hash 校验和    │2作为对称算法的密钥校验和    │3作为RSA密钥的公钥部分(最大2048 bit)    │├UINT16 size    │└BYTE buffer[256]    └4作为椭圆曲线坐标点x,y |
| name | 【TPM2B\_NAME】 | 当前节点的节点名,内部数据以二进制形式存储,包含三部分:   1. 数据块总长度。 2. 两字节哈希算法代码。 3. 由该哈希算法计算而得的摘要,字节数等于摘要默认长度。   其中第2、3部分数据被直接捆绑在一起  **name**  ├size：  └name：例如哈希算法代码0x0004 + 20字节SHA1哈希摘要 |
| qualifiedName | 【TPM2B\_DIGEST】 | Qualified Name由父节点的哈希值+当前节点的Name然后进行哈希运算而得,缩写为QN  QN子节点 = Hash子节点(QN父节点 + Name子节点)  特殊情况:密钥树主节点的父节点是根种子, 由于种子本身没有节点名Name和QN, 所以计算时取根种子的Handle代替其QN, 公式变为：  QNPrimary节点 = Hash(Handle父节点 + Name节点) |
| creationData |  | 创建一个普通节点时自动保存若干信息：包括PCR摘要、locality、父节点的哈希算法代码、父节点的QualifiedName、用户提供的额外附加数据 |

* + 1. 密钥对象的创建、密钥树节点的加载

**函数名：**

Tss2\_Sys\_Create()

Tss2\_Sys\_Load()

**功能描述：**

函数 Tss2\_Sys\_Create(): 让 TPM 在其内部创建一个密钥对象，注意 Create 命令并不会立即创建密钥树中相应的节点。密钥对象只有被加载之后才能用于其他运算。

函数 Tss2\_Sys\_Load(): 将 TPM 内部已经存在的密钥对象加载到密钥树中，作为父节点之下的一个子节点, 同时返回一个32位整数句柄。后续程序可凭此句柄选中密钥进行其他密码学运算。

**函数名：**

Tss2\_Sys\_SaveContext()

Tss2\_Sys\_LoadContext()

Tss2\_Sys\_FlushContext()

**功能描述：**

函数 Tss2\_Sys\_SaveContext(): 创建节点快照，保存当前密钥对象的运行时数据。 组合使用SaveContext/FlushContext/LoadContext 三个命令可以将暂时不用的密钥节点进行备份，然后从 TPM 运行时空间移出，腾出位置进行其他操作。

函数 Tss2\_Sys\_FlushContext(): 用于销毁密钥对象，即使密钥对象当前已经被加载到密钥树节点，这个节点也会被一并删除。节点被删除之后原先的句柄也失效。

函数 Tss2\_Sys\_LoadContext(): 通过快照恢复之前的密钥节点。通过该命令恢复的子节点是独立存在的，无论其父节点是否已经被删除，都能够恢复至创建节点快照时状态。

**接口定义：**

TPM\_RC Tss2\_Sys\_ContextSave(

TSS2\_SYS\_CONTEXT \*sysContext, // 上下文

TPMI\_DH\_CONTEXT saveHandle, // 输入

TPMS\_CONTEXT \*context // 输出

);

TPM\_RC Tss2\_Sys\_FlushContext(

TSS2\_SYS\_CONTEXT \*sysContext, // 上下文

TPMI\_DH\_CONTEXT flushHandle // 输入

);

TPM\_RC Tss2\_Sys\_ContextLoad(

TSS2\_SYS\_CONTEXT \*sysContext, // 上下文

TPMS\_CONTEXT \*context, // 输入

TPMI\_DH\_CONTEXT \*loadedHandle // 输出

);

**接口定义：**

TPM\_RC Tss2\_Sys\_Create(

TSS2\_SYS\_CONTEXT \*sysContext, // 上下文

TPMI\_DH\_OBJECT parentHandle, // 输入

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //

TPM2B\_SENSITIVE\_CREATE \*inSensitive, // 输入

TPM2B\_PUBLIC \*inPublic, /输入

TPM2B\_DATA \*outsideInfo, // 输入

TPML\_PCR\_SELECTION \*creationPCR, // 输入

**TPM2B\_PRIVATE** \*outPrivate, // 输出

**TPM2B\_PUBLIC** \*outPublic, // 输出

TPM2B\_CREATION\_DATA \*creationData, // 输出

TPM2B\_DIGEST \*creationHash, // 输出

TPMT\_TK\_CREATION \*creationTicket, // 输出

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

);

TPM\_RC Tss2\_Sys\_Load(

TSS2\_SYS\_CONTEXT \*sysContext, // 上下文

TPMI\_DH\_OBJECT parentHandle, // 输入

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //

**TPM2B\_PRIVATE** \*inPrivate, // 输入

**TPM2B\_PUBLIC** \*inPublic, // 输入

TPM\_HANDLE \*objectHandle, // 输出

TPM2B\_NAME \*name, // 输出

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

);

**输入输出参数：**

| Create命令  输入/输出参数 | 数据结构名称 | 数据结构示意图 |
| --- | --- | --- |
| 输出参数  outPrivate | 【TPM2B\_PRIVATE】/【\_PRIVATE】  【TPM2B\_SENSITIVE】/【TPMT\_SENSITIVE】  密钥类型**sensitiveType:**【TPMI\_ALG\_PUBLIC】  取值范围：  0x0001 TPM\_ALG\_RSA  0x0023 TPM\_ALG\_ECC  0x0025 TPM\_ALG\_SYMCIPHER  对称算法也可以是  0X0013 TPM\_ALG\_SM4  私钥数据格式:  【TPMU\_SENSITIVE\_COMPOSITE】  其中包括四种常见私钥格式  【TPM2B\_PRIVATE\_KEY\_RSA】  【TPM2B\_ECC\_PARAMETER】  【TPM2B\_SENSITIVE\_DATA】  【TPM2B\_SYM\_KEY】  和厂商自定义格式  【TPM2B\_PRIVATE\_VENDOR\_SPECIFIC】 | **outPrivate**  ├UINT16 size 数据块长度  └BYTE buffer[]  buffer内部的内容是变长数据流：   ├**integrityOuter:** 哈希摘要   │├UINT16 size   │└BYTE buffer[20～64字节变长数组]   ├**integrityInner:** 哈希摘要   │├UINT16 size   │└BYTE buffer[20～64字节变长数组]   │   └**sensitive**    ├UINT16 size    └**sentitiveArea**     ├**sensitiveType:** 2字节,密钥类型取值     ├**authValue**     │├UINT16 size     │└BYTE buffer[20～64字节变长数组]     ├**seedValue**     │├UINT16 size     │└BYTE buffer[20～64字节变长数组]     │     └**sensitive:** 记录不同类型密钥的原始值, 最大642字节      │1、rsa: RSA私钥数值      │ ├UINT16 size      │ └BYTE buffer[MAX\_RSA\_KEY\_BYTES/2＝128字节]      │2、ecc: 椭圆曲线的私钥整数数值, 最大48字节      │ ├UINT16 size      │ └BYTE buffer[MAX\_ECC\_KEY\_BYTES]      │3、bits: 1024 bit(128字节)隐私数据      │ ├UINT16 size      │ └BYTE buffer[]      │4、sym: 记录对称密钥的数值      │ ├UINT16 size      │ └BYTE buffer[16或32字节]      └其他厂商自定义的格式        ├UINT16 size        └BYTE buffer[最大长度640字节]           (640=5\*MAX\_RSA\_KEY\_BYTES/2) |
| 输入参数  inSensitive | 【TPM2B\_SENSITIVE\_CREATE / TPMS\_SENSITIVE\_CREATE】  【TPM2B\_AUTH】  【TPM2B\_DATA】 | **inSensitive**  │  ├UINT16 size 数据块长度(作为输入参数时开发者无需手动填写size数值)  └**sensitive**   │   ├**userAuth**：授权校验值,实际长度不能超过nameAlg哈希摘要长度   │├UINT16 size (必填)   │└BYTE buffer[]   │   └**data**：只在创建纯数据节点时用于指定数据内容。 一般可被省略    ├UINT16 size    └UINT8 buffer[] |

| Load命令  输入/输出参数 | 数据结构名称 | 数据结构示意图 |
| --- | --- | --- |
| 输入参数  inPrivate |  | **inPrivate**  ├UINT16 size 数据块长度①  └BYTE buffer[sizeof(\_PRIVATE)]   1. : 此处size 必须与 Create 命令所返回的原始数值一致 |
| 输入参数  inPublic | inPublic：  【TPM2B\_PUBLIC】  密钥类型选择  publicArea.**type**：  【TPMI\_ALG\_PUBLIC】  取值范围：  0x08TPM\_ALG\_KEYEDHASH  0x25 TPM\_ALG\_SYMCIPHER  0x01TPM\_ALG\_RSA  0x23TPM\_ALG\_ECC  0x13 TPM\_ALG\_SM4  密钥树节点名称选择哈希散列方式 publicArea.**nameAlg**：  【TPMI\_ALG\_HASH】  取值范围：  0x0010 TPM\_ALG\_NULL  0x0004 TPM\_ALG\_SHA1  0x000B TPM\_ALG\_SHA256  0x000C TPM\_ALG\_SHA384  0x000D TPM\_ALG\_SHA512  0x0012 TPM\_ALG\_SM3\_256  密钥的详细参数分5种情况  **parameters**  【TPMU\_PUBLIC\_PARMS】  (比如RSA密钥内部使用AES的ECB计算模式)  parameters.**rsaDetail**：  【TPMS\_RSA\_PARMS】  0x0006TPM\_ALG\_AES  0x0044 TPM\_ALG\_ECB | **inPublic**  ├UINT16 size(注：目前TSS内部会忽略此处size字段的数值，  │ 并依次读取下列数据块，然后统计压缩(Marshal)后总字节数)  │  └**publicArea**   │   ├**type**=密钥类型(例如：TPM\_ALG\_RSA)   │   ├**nameAlg**=指定哈希算法名(例如：TPM\_ALG\_SHA1)   │   ├**objectAttributes**=属性定义由32个标识位组成   │├1)密钥用途：签名/加解密/受限制;   │├2)是否要求授权访问(userWithAuth, adminWithPolicy, noDA);   │├3)duplication (fixedParent, fixedTPM,encryptedDuplication);   │├4)sensitiveDataOrigin   │└5)密钥是否会被清除(stClear);   │   ├**authPolicy**=授权策略的校验值,当adminWithPolicy属性标识位被置1时有效   │├UINT16 size   │└BYTE buffer[]   │   ├**parameters**=密钥细节参数,具体解析格式分下列5种情况   ││1作为keyed hash参数**keyedHashDetail**   ││2作为对称密钥参数 symDetail   ││3非对称密钥：作为RSA密钥的详细参数**rsaDetail**   ││4非对称密钥：作为椭圆曲线密钥的详细参数eccDetail   │└5作为非对称密钥通用参数 asymDetail   │   └**unique**=特征码,解析格式分下列4种情况    │1作为keyed hash 校验和    │2作为对称算法的密钥校验和    │3作为RSA密钥的公钥部分(最大2048 bit)    │├UINT16 size    │└BYTE buffer[256]    └4作为椭圆曲线坐标点x,y  rsaDetail中包括签名模式 signing scheme 设置参数 |
|  | 【TPM2B\_DATA】 | 暂时没有用到 |
|  | 【TPML\_PCR\_SELECTION】 | 选择PCR |

**函数名：**

Tss2\_Sys\_LoadExternal()

**功能描述：**

加载外部密钥。具体分下列三种常见情况：

* 只导入非对称密钥的公钥部分(不导入私钥部分)，可用于校验其他发信人的非对称数字签名(只需要导入发信人的公钥)。
* 导入一个对称密钥，然后调用 TPM 的对称算法引擎处理其他数据。
* 同时导入非对称密钥的公钥和私钥部分，并把 TPM 作为一个硬件加速器来使用。这种情况实际上并不是为了加快密码计算的速度，而只为了是让客户 App 更好地专注于处理自己业务逻辑，不需要客户自己去编码底层的加密算法逻辑（前提是客户 App 需要引入的 TSS 接口本身的逻辑，但 TPM 本身的工作逻辑比较负责复杂，有时可能会得不偿失)。另外，这种情况只能把密钥节点加载为 TPM\_RH\_NULL 树的主节点，而不能加载到其他节点之下，也不能在当前节点下创建其他子节点。

**接口定义：**

TPM\_RC Tss2\_Sys\_LoadExternal(

TSS2\_SYS\_CONTEXT\*sysContext, // IN

TSS2\_SYS\_CMD\_AUTHS const\*cmdAuthsArray, // IN

TPM2B\_SENSITIVE \*inPrivate, // IN

TPM2B\_PUBLIC \*inPublic, // IN

TPMI\_RH\_HIERARCHY hierarchy, // IN

TPM\_HANDLE \*objectHandle, // OUT

TPM2B\_NAME \*name, // OUT

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray // OUT

);

**输入输出参数：**

LoadExternal的输入输出参数

| 输入/输出  参数 | 数据结构名称 | 数据结构示意图 |
| --- | --- | --- |
| 输入参数  inPrivate | 【TPM2B\_SENSITIVE】 /【TPMT\_SENSITIVE】  密钥类型：**sensitiveType**【TPMI\_ALG\_PUBLIC】取值范围：  0x0001 TPM\_ALG\_RSA  0x0023 TPM\_ALG\_ECC  0x0025 TPM\_ALG\_SYMCIPHER  私钥数据:**sensitive**  【TPMU\_SENSITIVE\_COMPOSITE】 | **inPrivate**    ├UINT16 size注：目前TSS内部会忽略此处size字段的数值，    │ 并会依次读取下列数据块，然后统计压缩(Marshal)后总字节数    │    └**sentitiveArea**     ├**sensitiveType:** 2字节,密钥类型     ├**authValue:**     │├UINT16 size     │└BYTE buffer[]     ├**seedValue**: 唯一随机数种子     │├UINT16 size     │└BYTE buffer[]     │     └**sensitive: 记录不同类型密钥的私钥数据**      │1、rsa: RSA私钥数值      │ ├UINT16 size      │ └BYTE buffer[]      │2、ecc: 椭圆曲线的私钥整数数值      │ ├UINT16 size      │ └BYTE buffer[]      │3、bits: 1024 bit(128字节)隐私数据      │ ├UINT16 size      │ └BYTE buffer[]      │4、sym: 记录对称密钥的数值      │ ├UINT16 size      │ └BYTE buffer[]      └其他厂商扩展的格式 |

**函数名：**

Tss2\_Sys\_CreatePrimary()

**功能描述：**

创建主节点并将该节点加载到指定的根节点下。

**接口定义：**

TPM\_RC Tss2\_Sys\_CreatePrimary(

TSS2\_SYS\_CONTEXT \*sysContext, // IN

TPMI\_RH\_HIERARCHY primaryHandle, // IN

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, // IN

TPM2B\_SENSITIVE\_CREATE \*inSensitive, // IN

TPM2B\_PUBLIC \*inPublic, // IN

TPM2B\_DATA \*outsideInfo, // IN

TPML\_PCR\_SELECTION \*creationPCR, // IN

TPM\_HANDLE \*objectHandle, // OUT

TPM2B\_PUBLIC \*outPublic, // OUT

TPM2B\_CREATION\_DATA \*creationData, // OUT

TPM2B\_DIGEST \*creationHash, // OUT

TPMT\_TK\_CREATION \*creationTicket, // OUT

TPM2B\_NAME \*name, // OUT

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray // OUT

);

**输入输出参数：**

| 上下文指针参数 | 类型 | 用途 |
| --- | --- | --- |
| sysContext | TSS2\_SYS\_CONTEXT  (指针) | System API 上下文 |

| 对象句柄 | 类型 | 对象用途以及句柄取值可选范围 |
| --- | --- | --- |
| primaryHandle | TPMI\_RH\_HIERARCHY | TPM\_RH\_NULL  TPM\_RH\_OWNER  TPM\_RH\_ENDORSEMENT  TPM\_RH\_PLATFORM |

| 通用参数 | 类型 | 用途 |
| --- | --- | --- |
| cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS | 输入参数，填写命令桢授权会话区数据 |
| rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS | 输出参数，取回应答桢授权会话授权结果 |

| CreatePrimary命令  输入参数 | 数据结构名称 | 数据结构示意图 |
| --- | --- | --- |
| 输入参数  inSensitive | 【TPM2B\_SENSITIVE\_CREATE / TPMS\_SENSITIVE\_CREATE】  【TPM2B\_AUTH】  【TPM2B\_DATA】 | **inSensitive**  │  ├UINT16 size ()  └**sensitive**   │   ├**userAuth**：授权校验值,实际长度不能超过nameAlg哈希摘要长度   │├UINT16 size (必填)   │└BYTE buffer[]   │   └**data**：只在创建纯数据节点时用于指定数据内容。 一般可被省略    ├UINT16 size    └UINT8 buffer[] |
| 输入参数  inPublic  此参数作为一个模板,最终由 TPM 设备根据此模板输出完整的结果 outPublic | inPublic：  【TPM2B\_PUBLIC】  密钥类型选择  publicArea.**type**：  【TPMI\_ALG\_PUBLIC】  取值范围：  0x08 TPM\_ALG\_KEYEDHASH  0x25 TPM\_ALG\_SYMCIPHER  0x01 TPM\_ALG\_RSA  0x23 TPM\_ALG\_ECC  //0x13 TPM\_ALG\_SM4  密钥树节点名称选择哈希散列方式 publicArea.**nameAlg**：  【TPMI\_ALG\_HASH】  取值范围：  0x0010 TPM\_ALG\_NULL  0x0004 TPM\_ALG\_SHA1  0x000B TPM\_ALG\_SHA256  0x000C TPM\_ALG\_SHA384  0x000D TPM\_ALG\_SHA512  0x0012 TPM\_ALG\_SM3\_256  密钥的详细参数分5种情况  **parameters**  【TPMU\_PUBLIC\_PARMS】  (比如RSA密钥内部使用AES的ECB计算模式)  parameters.**rsaDetail**：  【TPMS\_RSA\_PARMS】  0x0006TPM\_ALG\_AES  0x0044 TPM\_ALG\_ECB | **inPublic**  │  ├UINT16 size (作为输入参数时开发者无需手动填写size数值)  └**publicArea**   │   ├**type**=密钥类型(例如：TPM\_ALG\_RSA)   │   ├**nameAlg**=指定哈希算法名(例如：TPM\_ALG\_SHA1)   │   ├**objectAttributes**=属性定义由32个标识位组成   │├1)密钥用途：签名/加解密/受限制;   │├2)是否要求授权访问(userWithAuth, adminWithPolicy, noDA);   │├3)duplication (fixedParent, fixedTPM,encryptedDuplication);   │├4)sensitiveDataOrigin   │└5)密钥是否会被清除(stClear);   │   ├**authPolicy**=授权策略的校验值,当adminWithPolicy属性标识位被置1时有效   │├UINT16 size   │└BYTE buffer[]   │   ├**parameters**=密钥细节参数,具体解析格式分下列5种情况   ││1作为keyed hash参数**keyedHashDetail**   ││2作为对称密钥参数 symDetail   ││3非对称密钥：作为RSA密钥的详细参数**rsaDetail**   ││4非对称密钥：作为椭圆曲线密钥的详细参数eccDetail   │└5作为非对称密钥通用参数 asymDetail   │   └**unique**=特征码,解析格式分下列4种情况    │1作为keyed hash 校验和    │2作为对称算法的密钥校验和    │3作为RSA密钥的公钥部分(最大2048 bit)    │├UINT16 size    │└BYTE buffer[256]    └4作为椭圆曲线坐标点x,y  rsaDetail中包括签名模式 signing scheme 设置参数 |
| outsideInfo | 【TPM2B\_DATA】 | 暂时没有用到 |
| creationPCR | 【TPML\_PCR\_SELECTION】 | 选择PCR |

| 应答桢  输出参数 | 数据结构名称 | 数据结构说明 |
| --- | --- | --- |
| objectHandle | 【TPM\_HANDLE】 | 返回本次成功创建的节点句柄编号 |
| outPublic  根据之前提供的模板, 由TPM创建而成 | 【TPM2B\_PUBLIC】 | **outPublic**  │  ├UINT16 size  └**publicArea**   │   ├**type**=密钥类型(例如：TPM\_ALG\_RSA)   │   ├**nameAlg**=指定哈希算法名(例如：TPM\_ALG\_SHA1)   │   ├**objectAttributes**=属性定义由32个标识位组成   │├a)密钥用途：签名/加解密/受限制   │├b)是否要求授权访问(userWithAuth, adminWithPolicy, noDA)   │├c)duplication (fixedParent, fixedTPM, encryptedDuplication)   │├d)sensitiveDataOrigin   │└e)密钥是否会被清除(stClear)   │   ├**authPolicy**=授权策略的校验值,当adminWithPolicy属性标识位被置1时有效   │├UINT16 size   │└BYTE buffer[]   │   ├**parameters**=密钥细节参数,具体解析格式分下列5种情况   ││1作为keyed hash参数   ││2作为对称密钥参数   ││3非对称密钥：作为RSA密钥的详细参数**rsaDetail**   ││4非对称密钥：作为椭圆曲线密钥的详细参数eccDetail   │└5作为非对称密钥通用参数   │   └**unique**=特征码,解析格式分下列4种情况    │1作为keyed hash 校验和    │2作为对称算法的密钥校验和    │3作为RSA密钥的公钥部分(最大2048 bit)    │├UINT16 size    │└BYTE buffer[256]    └4作为椭圆曲线坐标点x,y |
| creationData | 【TPM2B\_CREATION\_DATA】 | 创建一个Primary节点时, TPM 要自动保存若干与创建环境相关的信息, 包括：   * 当时所选PCR寄存器中的摘要数据 * locality 数值 * 父节点的哈希算法代码(因根节点没有哈希算法，故取 TPM\_ALG\_NULL) * 父节点的 Name (根种子没有节点名则取其句柄作为代替) * 父节点的 Qualified Name (根种子没有QN则取其句柄作为代替) * 调用者额外附加的其他数据 |
| creationHash | 【TPM2B\_DIGEST】 | creationData的哈希校验和 |
| creationTicket | 【TPMT\_TK\_CREATION】 | TPM自身对 creationData 进行签名(后续操作中, TPM 校验数据完整性时用到)  数据结构：  **creationTicket**  ├tag：  ├hierarchy：  └**digest**：HMAC输出的摘要 |
| name | 【TPM2B\_NAME】 | 当前节点的节点名,内部数据以二进制形式存储,包含三部分:   1. 数据块总长度。 2. 两字节哈希算法代码。 3. 由该哈希算法计算而得的摘要,字节数等于摘要默认长度。   其中第2、3部分数据被直接捆绑在一起  **name**  ├size：  └name：例如哈希算法代码0x0004 + 20字节SHA1哈希摘要 |

**System API 返回码：**

TSS2\_RC：32位无符号整数，错误编码的含义如下表：

| 返回码 | 返回码的宏定义 | 含义 |
| --- | --- | --- |
| 0x0 | TSS2\_RC\_SUCCESS | TSS接口API函数执行成功 |
| 0x0008000B | TSS2\_SYS\_RC\_BAD\_VALUE | 某一个输出参数的初始值无效。  备注：输出参数 outPublic 和 creationData 的 size 初始值应等于0，否则 Tss2\_Sys\_CreatePrimary() 将返回该错误码 |
|  |  |  |

**TPM 返回码：**

TPM\_RC：32位无符号整数，错误编码的含义如下表：

| 返回码 | 返回码的宏定义 | 含义 |
| --- | --- | --- |
| 0x0 | TPM\_RC\_SUCCESS | 当前TPM命令执行成功 |
| 0x80|0x02 | TPM\_RC\_ATTRIBUTES | 如果产生该错误码，有如下几种可能的情况：  sensitiveDataOrigin is CLEAR when 'sensitive.data' is an Empty Buffer, or is SET when 'sensitive.data' is not empty;  fixedTPM, fixedParent, or encryptedDuplication attributes are inconsistent between themselves or with those of the parent object;  inconsistent restricted, decrypt and sign attributes; attempt to inject sensitive data for an asymmetric key;  attempt to create a symmetric cipher key that is not a decryption key |
| 0x80|0x0C | TPM\_RC\_KDF | incorrect KDF specified for decrypting keyed hash object |
| 0x80|0x1C | TPM\_RC\_KEY | a provided symmetric key value is not allowed |
| 0x0902 | TPM\_RC\_OBJECT\_MEMORY | 表示 TPM 当前加载的(或已创建但未加载的)密钥总数量已经达到可用空间的上限, 无法继续创建新的密钥。 |
| 0x80|0x012 | TPM\_RC\_SCHEME | inconsistent attributes decrypt, sign, restricted and key's scheme ID; or hash algorithm is inconsistent with the scheme ID for keyed hash object |
| 0x80|0x15 | TPM\_RC\_SIZE | size of public authorization policy or sensitive authorization value does not match digest size of the name algorithm; or sensitive data size for the keyed hash object is larger than is allowed for the scheme |
| 0x80|0x16 | TPM\_RC\_SYMMETRIC | a storage key with no symmetric algorithm specified; or non-storage key with symmetric algorithm different from TPM\_ALG\_NULL |
| 0x80|0x0A | TPM\_RC\_TYPE | unknown object type |

**函数名：**

Tss2\_Sys\_CreateLoaded()

**功能描述：**

创建一个普通节点并立即将该节点加载到指定的父节点下, 如果父节点是 TPM 某个根节点, 则实际效果与函数 Tss2\_Sys\_CreatePrimary() 完全相同。

**TPM 的三种 Create 命令之间的对比：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Create命令** | TPM2\_CreatePrimary | TPM2\_Create | TPM2\_CreateLoaded | | |
| **软件栈API接口函数** | Tss2\_Sys\_CreatePrimary() | Tss2\_Sys\_Create() | Tss2\_Sys\_CreateLoaded() | | |
| **父节点位置** | Primary Seed | Storage Parent | PrimarySeed | Storage Parent | Derivation Parent |
| **节点类型** | 主节点 | 普通节点 | 主节点 | 普通节点 | Derived |
| **TPM 应答桢是否返回public area** | yes | yes | yes | yes | yes |
| **TPM 应答桢是否返回sensitive area** | no | yes | no | yes | no |
| **TPM 应答桢是否返回creation data** | yes | yes | no | no | no |

|  |  |
| --- | --- |
| 命令 | 说明 |
| **TPM2\_CreatePrimary** | Creates and loads Primary Objects for immediate use, and provides creationData. |
| **TPM2\_Create** | Creates Ordinary Objects for later use (via TPM2\_Load). TPM2\_Create returns a BLOB containing the sensitive area of an Ordinary Object and provides creationData. |
| **TPM2\_CreateLoaded** | Depending on the type of the parent, generates and loads a Primary Object, an Ordinary Object; or Derived Object. |

1. 其他命令
   1. 其他命令
      1. Tss2\_Sys\_EvictControl

**函数名：**Tss2\_Sys\_EvictControl（）

**功能描述：**此命令允许某些瞬态对象备份成持久化对象或持久对象被驱逐。

**接口定义：**

TPM\_RC Tss2\_Sys\_EvictControl(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION auth,

TPMI\_DH\_OBJECT objectHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPMI\_DH\_PERSISTENT persistentHandle,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

当objectHandle指向瞬态对象时：

1. TPM\_RC\_ATTRIBUTES 如果：
2. Hierarchy为TPM\_RH\_NULL，或者
3. 只有对象的公共部分被加载，或者
4. 对象的stClear被置位或in an ancestor key。
5. TPM\_RC\_HIERARCHY， auth的类型应当与objectHandle定义的类型相同，否则返回该错误。
6. objectHandle如果为Platform hierarchy，则auth为TPM\_RH\_PLATFORM
7. objectHandle如果为Platform hierarchy，如果auth为TPM\_RH\_OWNER，则正确的hierarchy应当是storage或Endorsement hierarchy。
8. TPM\_RC\_RANGE 如果persistentHandle不在auth确定的正确范围内，则返回该错误。
9. 如果auth为TPM\_RH\_OWNER，则正确的persistentHandle应当是0x81000000 到

0x817FFFFF 。

1. 如果auth为TPM\_RH\_PLATFORM，则正确的persistentHandle应当是0x81800000到

0x81FFFFFF。

1. TPM\_RC\_NV\_DEFINED 如果持久对象persistentHandle已经存在，则返回该错误。
2. TPM\_RC\_NV\_SPACE 如果没有足够的空间创建持久对象，则……
3. TPM\_RC\_NV\_SPACE 如果执行此命令将阻止TPM能够容纳任何类型的两个临时对象，TPM将返回该错误。
4. 如果TPM返回TPM\_RC\_SUCCESS，则objectHandle引用的对象不会被清空，并且可以使用objectHandle和persistentHandle来访问该对象。

当objectHandle指向持久对象时：

1. TPM\_RC\_RANGE 如果objectHandle没有在auth确定的范围内时，返回该错误。

Auth为TPM\_RC\_OWNER，objectHandle的范围为0x81000000到0x817FFFFF

Auth为TPM\_RC\_PLATFORM，objectHandle可以为任意有效的对象句柄。

1. TPM\_RC\_HANDLE 如果objectHandle的值与persistentHandle不同时，将返回该错误。
2. TPM\_RC\_SUCCESS objectHandle将会从持久内存中移除，并不在可用。

注：

1. 如果objectHandle为瞬态对象，则该命令将会创建一个持久备份并将persistentHandle设置为该持久对象的句柄。如果objectHandle为持久对象，则该命令将会驱逐该持久对象。
2. 该命令并不对瞬态对象产生影响。
3. 在执行该命令之前，会先验证objectHandle是TPM中存在的handle，并且persistentHandle是一个无效的handle。
4. 如果TPM返回TPM\_RC\_SUCCESS，则objectHandle引用的对象不会被清空，并且可以使用objectHandle和persistentHandle来访问该对象。
5. PCR相关命令
   1. PCR相关命令
      1. PCR Allocate

**函数名：**Tss2\_Sys\_PCR\_Allocate（）

**功能描述：**给给定的PCRs分配摘要算法。将每个bank下的pcr的bitmap等信息，记录到全局变量中。

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_Allocate(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TPMI\_RH\_PLATFORM authHandle, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

TPML\_PCR\_SELECTION \*pcrAllocation, //in

TPMI\_YES\_NO \*allocationSuccess, //out

UINT32 \*maxPCR, //out

UINT32 \*sizeNeeded, //out

UINT32 \*sizeAvailable, //out

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray //out

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 结构 |
| sysContext | TSS2\_SYS\_CONTEXT \* | System API 上下文 |
| authHandle | TPMI\_RH\_PLATFORM | 指定授权的用户TPM\_RH\_PLATFORM |
| cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS const \* | TSS2\_SYS\_CMD\_AUTHS  |-- uint8\_t cmdAuthsCount授权区域个数  |--\*\*cmdAuths  |--sessionHandle； 会话句柄  |--nonce  |--sessionAttributes  |--hmac |
| pcrAllocation | TPML\_PCR\_SELECTION\* | TPML\_PCR\_SELECTION  |-- count; 将要修改的pcrbank的个数  |--pcrSelection；要修改bank中pcr的信息  |--hash；算法（eg：TPM\_ALG\_SHA256）  |--sizeofSelect；bitmap所占字节数；  |--pcrSelect[PCR\_SELECT\_MAX];  选择的pcr的bit map |
| allocationSuccess | TPMI\_YES\_NO \* | 输出参数。  TPM\_RC\_PCR：将要分配的PCR被DRTM或HCRTM占用。  TPM\_RC\_SUCCESS：成功 |
| sizeNeeded | UINT32 \* | 分配pcr需要使用的大小 |
| sizeAvailable | UINT32 \* | 可用大小 |
| rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS | 输出参数同输入参数 |

* + 1. PCR Extend

**函数名：**Tss2\_Sys\_PCR\_Extend ()

**功能描述：**根据digests中的bank数据，对指定的PCR数据按照传入的hash算法进行下的extend。

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_Extend(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TPMI\_DH\_PCR pcrHandle, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

TPML\_DIGEST\_VALUES \*digests, //in

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray //out

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| pcrHandle | TPMI\_DH\_PCR | 需要extend的PCR |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS const \* | 授权信息 |
| \*digests | TPML\_DIGEST\_VALUES \* | 将要进行扩展的字符串  TPML\_DIGEST\_VALUES  |---count：扩展bank的个数  |---TPMT\_HA  |---hashAlg 指定的bank算法  |---TPMU\_HA hash值 |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS \* | 输出参数同输入参数 |

**举例：**

修改bank1中pcr的值：

其中digests.digests[0].digest.sha1值为

000102030405060708090a0b0c0d0e0f10111213时，如果此时将要进行PCR17扩展的PCR17的值为：34d4b55ae22c7dd4313f272b114e2c33422e4cf4。

Extend函数的结果就是将要进行扩展的字符串放在PCR的值后面进行扩展，然后再根据当前PCR的bank的算法，进行hash计算。

即前半段为PCR的值，后半部分为将要扩展的字符串：34d4b55ae22c7dd4313f272b114e2c33422e4cf4000102030405060708090a0b0c0d0e0f10111213

如果对bank0（sha1）中的PCR的值进行hash算法，将会得到与用sha1处理上述字符串相同的值912b806d13a90982b8a87a619b4053925881bf58。

* + 1. PCR Read

**函数名：**Tss2\_Sys\_PCR\_Read ()

**功能描述：**读取pcrSelectionIn给定的值。

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_Read(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

TPML\_PCR\_SELECTION \*pcrSelectionIn, //in

UINT32 \*pcrUpdateCounter, //out

TPML\_PCR\_SELECTION \*pcrSelectionOut, //out

TPML\_DIGEST \*pcrValues, //out

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray //out

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS const \* | 授权信息 |
| \*pcrSelectionIn | TPML\_PCR\_SELECTION \* | 将要读取的PCR的bitmap |

**输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*pcrUpdateCounter | UINT32 | 返回Pcr改变的计数 |
| \*pcrSelectionOut | TPML\_PCR\_SELECTION\* | 同输入参数pcrSelectionIn |
| \* pcrValues | TPML\_DIGEST \* | 选中的pcr的值 |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS \* | 输出参数同输入参数 |

* + 1. PCR Event

**函数名：**Tss2\_Sys\_PCR\_Event ()

**功能描述：**首先先对传入的eventData按照各个bank相对应的的算法进行hash摘要，然后再在对应的bank内的对传入的PCR用对应bank下的eventData的摘要值进行Extend。

简述：Event函数会将传入PCR的所有bank下的值进行extend。Extend的参数是EventData的hash摘要值。

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_Event(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TPMI\_DH\_PCR pcrHandle, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

TPM2B\_EVENT \*eventData, //in

TPML\_DIGEST\_VALUES \*digests,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

返回参数错误码：

TPM\_RC\_LOCALITY 0x907 //bad locality

TPM\_RC\_SUCCESS //成功

过程描述：

eventData中buffer为传入参数。该函数先对Buffer中的值进行hash摘要，然后在将摘要值Extend到指定的PCR，然后在该PCR所在的bank中都进行该操作。

Eg：

PCR18（sha1）值为：35ab992598f6baf2578ee68eb04265b713379718

eventData中buffer大小为4， buffer值为00ff55aa，

PCR18

00ff55aa

EventData

2ddd5b98e20956cdd58

81fa80fdf7c2aa7da7278

SHA1

35ab992598f6baf2578ee68eb04265b713379718

a6a0cdd21a1e742a5ffef

673d5666cdcd5fb38e0

Extend

Bank1(sha1)

Bank1

PCR18

Bank1

Bank2

Bank n

……

……

……

SHA1

SHA256

……

EventData

(Event)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 参数 |
| \*sysContext | TSS2\_SYS\_CONTEXT\* | 上下文 |
| pcrHandle | TPMI\_DH\_PCR | 要操作的PCR |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |
| \*eventData | TPM2B\_EVENT\* | 输入数据  typedef union {  struct {  UINT16 size;  BYTE buffer[1024];  } t;  TPM2B b;  } TPM2B\_EVENT; |

**输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 参数 |
| \*digests | TPML\_DIGEST\_VALUES\* | 对eventData进行哈希计算的结果 |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS\* | 输出授权信息 |

* + 1. PCR SetAuthPolicy

**函数名：**Tss2\_Sys\_PCR\_SetAuthPolicy ()

**功能描述：**设置PCR或者PCR组策略。（该策略决定在什么情况下进行pcr 扩展或复位。）

现在simlator代码不支持PCR策略。

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_SetAuthPolicy(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PLATFORM authHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //授权信息

TPM2B\_DIGEST \*authPolicy, //该值将最终写入到NV中

TPMI\_ALG\_HASH hashAlg, //该policy支持的算法

TPMI\_DH\_PCR pcrNum, //pcr的句柄值

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray //out

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| authHandle | TPMI\_RH\_PLATFORM | 授权者 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS | 授权信息 |
| \*authPolicy | TPM2B\_DIGEST | 写入到NV中NvWriteReserved |
| hashAlg | TPMI\_ALG\_HASH | 该policy对应的hash算法 |
| pcrNum | TPMI\_DH\_PCR | Pcr句柄 |

**返回值：**

Return：TPM\_RCS\_VALUE + RC\_PCR\_SetAuthPolicy\_pcrNum; （0x3c4）

* + 1. PCR SetAuthValue

**函数名：**Tss2\_Sys\_PCR\_SetAuthValue ()

**功能描述：**更改一个（或一组）PCR的授权值。

**接口定义：**TPM\_RC Tss2\_Sys\_PCR\_SetAuthValue(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TPMI\_DH\_PCR pcrHandle, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,//in

TPM2B\_DIGEST \*auth, //in

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT | 上下文 |
| pcrHandle | TPMI\_DH\_PCR | 要修改授权值的PCR |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS const | 授权信息 |
| \*auth | TPM2B\_DIGEST | 要修改的组策略信息  typedef union {  struct {  UINT16 size;  BYTE buffer[sizeof(TPMU\_HA)];  } t;  TPM2B b;  } TPM2B\_DIGEST; |

**返回值：**

Return：TPM\_RC\_VALUE（0x84） 该PCR没有组策略或不支持组策略。

TPM\_RC\_SUCCESS

输入参数结构体:

typedef union {

struct {

UINT16 size;

BYTE buffer[sizeof(TPMU\_HA)];

} t;

TPM2B b;

} TPM2B\_DIGEST;

注：

现在simlator代码不支持PCR策略。所以一直返回失败TPM\_RC\_VALUE。

* + 1. PCR Reset

**函数名：**TPM\_RC Tss2\_Sys\_PCR\_Reset ()

**功能描述：**当PCR的属性允许reset且提供正确的授权后，该命令将会把所有bank下的PCR的值设置成0。（reset操作收到PCR的resetLocality位属性限制，只有在允许的locality下才行进行该PCR的复位）

**接口定义：**

TPM\_RC Tss2\_Sys\_PCR\_Reset(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_DH\_PCR pcrHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT\* | 上下文 |
| pcrHandle | TPMI\_DH\_PCR | 要重置的PCR |
| \* cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |

**返回值：**

TPM\_RC\_LOCALITY(0x907) /\* bad locality \*/

1. 随机数相关命令
   1. 随机数相关命令
      1. GetRandom

**函数名：**Tss2\_Sys\_GetRandom（）

**功能描述：**在simulator下，该函数为OpenSSL 的RAND\_bytes（）函数产生的伪随机数。

**接口定义：**

TPM\_RC Tss2\_Sys\_GetRandom(

TSS2\_SYS\_CONTEXT \*sysContext, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

UINT16 bytesRequested, //in

TPM2B\_DIGEST \*randomBytes, //in

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 产生随机数不需要该参数，传NULL即可 |
| bytesRequested | UINT16 | 要产生随机数的字节数 |
| \* randomBytes | TPM2B\_DIGEST \* | 产生的随机数 |

输入参数结构：

typedef union {

struct {

UINT16 size;

BYTE buffer[sizeof(TPMU\_HA)];

} t;

TPM2B b;

} TPM2B\_DIGEST;

* + 1. StirRandom

**函数名：**Tss2\_Sys\_StirRandom（）

**功能描述：**在simulator下，该函数为OpenSSL 的RAND\_add（）函数将indata中的数据增加到RNG的state中，增加随机数的不可预测性。

**接口定义：**

TPM\_RC Tss2\_Sys\_StirRandom(

TSS2\_SYS\_CONTEXT \*sysContext,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_SENSITIVE\_DATA \*inData,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 产生随机数不需要此参数，传NULL即可。 |
| \* inData | TPM2B\_SENSITIVE\_DATA \* | 要增加的信息。 |

输入参数结构：

typedef union {

struct {

UINT16 size;

BYTE buffer[MAX\_SYM\_DATA];

} t;

TPM2B b;

}TPM2B\_SENSITIVE\_DATA;

1. Clock相关命令
   1. Clock相关命令
      1. Read clock

**函数名：**Tss2\_Sys\_\_ReadClock（）

**功能描述：**读取TPM的时钟信息、reset次数、重启次数等。

**接口定义：**

TPM\_RC Tss2\_Sys\_ReadClock(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMS\_TIME\_INFO \*currentTime

)

**输入参数：**sysContext 上下文

**输出参数：**TPMS\_TIME\_INFO \*currentTime

TPMS\_TIME\_INFO

|---- U64 time /\*time in milliseconds since the last \_TPM\_Init or TPM2\_Startup\*/

|----clockInfo

|-------U64 time

|-------resetCount

|-------restartCount

|-------BYTE safe /\* no value of Clock greater than the current value of Clock has been previously reported by the TPM.

* + 1. ClockSet

**函数名：**Tss2\_Sys\_ ClockSet（）

**功能描述：**读取TPM的时钟信息、reset次数、重启次数等。

**接口定义：**

TPM\_RC Tss2\_Sys\_ClockSet(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION auth,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

UINT64 newTime,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \* sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| auth | TPMI\_RH\_PROVISION | 需要平台或用户授权 |
| \* cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |
| newTime | UINT64 | 将要设置的时间 |

**参数取值范围：**

TPMI\_RH\_PROVISION

TPM\_RH\_OWNER ：用户授权

TPM\_RH\_PLATFORM ：平台授权

newTime：

要求将要设置的时间值必须要大于当前时间，并且要小于0xFFFF000000000000ULL。

**返回值：**

0x1C4: TPM\_RCS\_VALUE + RC\_ClockSet\_newTime;(输入的newTime时间参数越界)

* + 1. ClockRateAdjust

**函数名：**Tss2\_Sys\_ ClockRateAdjust（）

**功能描述：**调整时钟和时间的Rate，这个比率的调整是相对于当前rate来说，并不是相对于名义上的rate。

**接口定义：**

TPM\_RC Tss2\_Sys\_ClockRateAdjust(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION auth,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM\_CLOCK\_ADJUST rateAdjust,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \* sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| auth | TPMI\_RH\_PROVISION | 需要平台或用户授权 |
| \* cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |
| rateAdjust | TPM\_CLOCK\_ADJUST | 时钟调整rate |

**参数取值范围：**

TPMI\_RH\_PROVISION

TPM\_RH\_OWNER ：用户授权

TPM\_RH\_PLATFORM ：平台授权

rateAdjust：

TPM\_CLOCK\_COARSE\_SLOWER

TPM\_CLOCK\_MEDIUM\_SLOWER

TPM\_CLOCK\_FINE\_SLOWER

TPM\_CLOCK\_NO\_CHANGE

TPM\_CLOCK\_FINE\_FASTER

TPM\_CLOCK\_MEDIUM\_FASTER

TPM\_CLOCK\_COARSE\_FASTER

**返回值：**

【规范】TPM\_RCS\_VALUE：(调整比率超出范围)

1. NV相关命令
   1. NV相关命令
      1. Tss2\_Sys\_NV\_DefineSpace

**函数名：**Tss2\_Sys\_NV\_DefineSpace（）

**功能描述：**define一个NV的index，使TPM保留空间来保存与NV Index相关联的数据。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_DefineSpace(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION authHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_AUTH \*auth,

TPM2B\_NV\_PUBLIC \*publicInfo,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 参数类型 | 参数结构和描述 |
| \* sysContext | TSS2\_SYS\_CONTEXT\* | 上下文 |
| authHandle | TPMI\_RH\_PROVISION | 授权信息  1：TPM\_RH\_PLATFORM 平台授权  2：TPM\_RH\_OWNER 用户授权 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | TSS2\_SYS\_CMD\_AUTHS  |-- uint8\_t cmdAuthsCount授权区域个数  |--\*\*cmdAuths  |--sessionHandle； 会话句柄  |--nonce  |---size  |--buffer  |--sessionAttributes  |---continueSession  |---auditExclusive  |---auditReset  |---reserved1  |---decrypt  |---encrypt  |---audit  |--hmac：TPM2B\_DIGEST  |---size  |---buffer[64] |
| \*auth | TPM2B\_AUTH \* | typedef TPM2B\_DIGEST TPM2B\_AUTH; TPM2B\_DIGEST  |----size  |----buffer[64] |
| \*publicInfo | TPM2B\_NV\_PUBLIC \* | typedef union {  struct {  UINT16 size;  TPMS\_NV\_PUBLIC nvPublic; } t;  TPM2B b;  } TPM2B\_NV\_PUBLIC;  TPMS\_NV\_PUBLIC  |---- nvIndex NV索引值  |--- nameAlg hash算法  |--- attributes  |--- TPM2B\_DIGEST  |---size  |----buffer[64]  |----dataSize |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS \* |  |

NV attributes：

typedef union {

struct {

unsigned int TPMA\_NV\_PPWRITE : 1; //1:平台授权可写

unsigned int TPMA\_NV\_OWNERWRITE : 1; //1:owner授权可写

unsigned int TPMA\_NV\_AUTHWRITE : 1; //1:写操作需要用户HMAC会话或密码会话授权

unsigned int TPMA\_NV\_POLICYWRITE : 1; //写操作需要用户提供策略会话授权

unsigned int TPM\_NT : 4; //index 类型【注】TPM不需要实现所有的TPM\_NT值

unsigned int reserved1 : 2;

unsigned int TPMA\_NV\_POLICY\_DELETE : 1; //1:需要使用TPM2\_NV\_UndefineSpaceSpecial

取消NV，并且需要平台授权，

0:使用TPM2\_NV\_UndefineSpace，并且提供正 确的授权

unsigned int TPMA\_NV\_WRITELOCKED : 1; //1:写锁，不能写。

unsigned int TPMA\_NV\_WRITEALL : 1; //1:只能全部写，不允许部分写。

unsigned int TPMA\_NV\_WRITEDEFINE : 1; //1: TPM2\_NV\_WriteLock可能被使用阻止同

一位置的写入，

0: TPMA\_NV\_WRITE\_STCLEAR也为0时，

不会阻塞后续写入

unsigned int TPMA\_NV\_WRITE\_STCLEAR : 1; //1: TPM2\_NV\_WriteLock可能被使用阻止同

一位置的写入，直到TPM\_reset或restart

0: TPMA\_NV\_WRITE\_STCLEAR也为0时，

不会阻塞后续写入。

unsigned int TPMA\_NV\_GLOBALLOCK : 1; //1: TPM2\_NV\_GlobalWriteLock执行成功后，该

NV将不能写入，直到TPM reset 或者restart.

0: TPM2\_NV\_GlobalWriteLock函数将不起作用.

unsigned int TPMA\_NV\_PPREAD : 1; //1:平台授权可读

unsigned int TPMA\_NV\_OWNERREAD : 1; //1:用户授权可读

unsigned int TPMA\_NV\_AUTHREAD : 1; //1:提供authValue可读

unsigned int TPMA\_NV\_POLICYREAD : 1; //1:满足authPolicy可读

unsigned int reserved2 : 5;

unsigned int TPMA\_NV\_NO\_DA : 1; //1: 授权失败不影响DA logic,当tpm处于lockout

状态时,授权会话不阻塞

0:授权失败将会时授权失败计数器增加,且当tpm处于

lockout状态时,此NV index的授权将不被允许.

unsigned int TPMA\_NV\_ORDERLY : 1; //1: 当执行有序关闭(tpm2\_shutdown)时,NV index的

state是唯一要保存的

0:当执行完update index命令后,Nv index的state同步 更新.

unsigned int TPMA\_NV\_CLEAR\_STCLEAR : 1; //1: 当tpm reset或restart时,TPMA\_NV\_WRITTEN 将被清空.

0: TPMA\_NV\_WRITTEN不改变当tpm restart时。

unsigned int TPMA\_NV\_READLOCKED : 1; //1:读操作将被阻塞，直到tpm reset或restart。

0:当提供合适授权，允许读操作。

unsigned int TPMA\_NV\_WRITTEN : 1; //1:该nv已经被写过。

unsigned int TPMA\_NV\_PLATFORMCREATE : 1; //1:指定define NV的是平台授权还是owner授权

unsigned int TPMA\_NV\_READ\_STCLEAR : 1; //1: TPM2\_NV\_ReadLock将被用于设置 TPMA\_NV\_READLOCKED

0: TPM2\_NV\_ReadLock将不起作用

};

UINT32 val;

} TPMA\_NV;

参数值：

TPM\_NT的值

TPM\_NT\_COUNTER,

TPM\_NT\_BITS,

TPM\_NT\_PIN\_FAIL,

TPM\_NT\_PIN\_PASS

**返回值：**

1. TPM\_RC\_ATTRIBUTES，下列情况将会返回该错误值

（1）nvIndexType TPM\_NT的值不支持。

（2）TPMA\_NV\_WRITTEN, TPMA\_NV\_READLOCKED, or TPMA\_NV\_WRITELOCKED被置位时。

（3）TPMA\_NV\_PPREAD, TPMA\_NV\_OWNERREAD,TPMA\_NV\_AUTHREAD, TPMA\_NV\_POLICYREAD中至少一个要被置为1，（即必须有一种读取该NV的方法），否则…

（4）TPMA\_NV\_PPWRITE, TPMA\_NV\_OWNERWRITE, TPMA\_NV\_AUTHWRITE, TPMA\_NV\_POLICYWRITE中至少一个被置位，（即必须有一种写该NV的方法），否则….

（5）TPMA\_NV\_CLEAR\_STCLEAR = 1时，TPM\_NT不能设置为TPM\_NT\_COUNTER,否则….

（6）如果latformAuth/platformPolicy用于授权，则TPMA\_NV\_PLATFORMCREATE需要被置位，否则….ownerAuth/ownerPolicy用于授权，TPMA\_NV\_PLATFORMCREATE必须要被清0，否则…..

（7）TPMA\_NV\_POLICY\_DELETE=1时，需要平台授权,且TPM2\_NV\_UndefineSpaceSpecial()函数要实现，否则…..

（8）TPM\_NT = TPM\_NT\_PIN\_FAIL时, TPMA\_NV\_NO\_DA必须要设置为1，否则…(代码未实现)

（9）TPM\_NT = TPM\_NT\_PIN\_FAIL or TPM\_NT\_PIN\_PASS时，TPMA\_NV\_PPWRITE, TPMA\_NV\_OWNERWRITE, or TPMA\_NV\_POLICYWRITE至少其中之一将被置位，否则…

TPMA\_NV\_AUTHWRITE 必须被CLEAR，否则…..

（10）TPM\_NT = TPM\_NT\_COUNTER时，若不支持TPM2\_NV\_Increment(),则返回……

（11）TPM\_NT = TPM\_NT\_COUNTER时，TPMA\_NV\_CLEAR\_STCLEAR ！= 1，否则则返回…（代码实现但标准没有）

（12）TPM\_NT = TPM\_NT\_BITS时，若不支持TPM2\_NV\_SetBits(),则返回……

（13）TPM\_NT = TPM\_NT\_EXTEND时，若不支持TPM2\_NV\_Extend(),则返回……

（14）TPMA\_NV\_CLEAR\_STCLEAR和TPMA\_NV\_WRITEDEFINE不同时置位，否则…（代码实现但标准没有）

1. TPM\_RC\_NV\_DEFINED： NV index已经存在
2. TPM\_RC\_SIZE，当TPM\_NT的值为以下情况时，会返回TPM\_RC\_SIZE
3. TPM\_RC\_NV\_UNINITIALIZED当成功执行完defineSpace，NV Index已经存在，但TPMA\_NV\_WRITTEN=0时，任何访问NV 数据的操作都将返回错误码TPM\_RC\_NV\_UNINITIALIZED
4. TPM\_RC\_NV\_SPACE当由于缺少资源导致命令失败，将返回错误码TPM\_RC\_NV\_SPACE

（1）当为TPM\_NT\_COUNTER, TPM\_NT\_BITS, TPM\_NT\_PIN\_FAIL,TPM\_NT\_PIN\_PASS时， publicInfo→dataSize必须为8，否则返回TPM\_RC\_SIZE。

（2）当为TPM\_NT\_EXTEND， publicInfo→dataSize需要与 publicInfo.nameAlg的摘要大小一致，否则返回TPM\_RC\_SIZE。

（3）当TPM\_NT= TPM\_NT\_ORDINARY时，publicInfo→dataSize<MAX\_NV\_INDEX\_SIZE(2048)时，否则返回TPM\_RC\_SIZE

* + 1. Tss2\_Sys\_NV\_UndefineSpace

**函数名：**Tss2\_Sys\_NV\_ UndefineSpace（）

**功能描述：**undefine指定的NV index。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_UndefineSpace(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_HANDLE. Nv index未定义
2. TPM\_RC\_NV\_AUTHORIZATION platform创建的NV, owner用户不能删除该NV；反之则可以。
3. TPM\_RC\_ATTRIBUTES 如果该nv有TPMA\_NV\_POLICY\_DELETE == 1,用该函数删除该nv则会返回TPM\_RC\_ATTRIBUTES错误。

注：

NV\_define时如果采用TPM\_RH\_PLATFORM授权方式，只能使用TPM\_RH\_PLATFORM来取消NV定义，而如果采用用户授权的方式定义NV，既可以使用TPM\_RH\_PLATFORM来取消NV定义，也可以采用TPM\_RH\_OWNER来取消NV定义。

* + 1. Tss2\_Sys\_NV\_UndefineSpaceSpecial

**函数名：**Tss2\_Sys\_NV\_ UndefineSpaceSpecial（）

**功能描述：**该函数用于删除platform创建的带有TPMA\_NV\_POLICY\_DELETE 属性的指定index的NV。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_UndefineSpace(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_HANDLE. Nv index未定义
2. TPM\_RC\_NV\_AUTHORIZATION platform创建的NV, owner用户不能删除该NV；反之则可以。
3. TPM\_RC\_ATTRIBUTES 如果TPMA\_NV\_POLICY\_DELETE和TPMA\_NV\_PLATFORMCREATE位被clear，则会返回TPM\_RC\_ATTRIBUTES错误。
   * 1. Tss2\_Sys\_NV\_Read

**函数名：**Tss2\_Sys\_NV\_ Read（）

**功能描述：**从NV空间中读取先前已经定义的NV的数据。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_Read(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

UINT16 size,

UINT16 offset,

TPM2B\_MAX\_NV\_BUFFER \*data,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入参数：**

Size : 将要读取的数据大小

Offset：读取的数据的offset

**输出参数：**

Data：中buffer成员即读出的NV的数据。

**返回值：**

1. TPM\_RC\_NV\_LOCKED TPMA\_NV\_READLOCKED=1时，会返回该错误。
2. TPM\_RC\_NV\_RANGE offset 和 size加起来的大小大于 nv中的数据总大小。
3. TPM\_RC\_VALUE offset的大小大于nv中的数据总大小，则返回该错误。
4. TPM\_RC\_NV\_UNINITIALIZED，NV index已经定义，但TPMA\_NV\_WRITTEN不等于1，则会返回TPM\_RC\_NV\_UNINITIALIZED错误。

注：

TPM2B\_MAX\_NV\_BUFFER \*data,为传出数据，但是在数据读取之前，需要先设置data内buffer的大小，即先为data->t.size赋值。

* + 1. Tss2\_Sys\_NV\_ Write

**函数名：**Tss2\_Sys\_NV\_Write（）

**功能描述：**将data中的数据的合并到nvIndex->data[offset]开始的位置，可以重复写同一NV。

**接口定义：**

TPM\_RC TPM\_RC Tss2\_Sys\_NV\_Write(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_MAX\_NV\_BUFFER \*data,

UINT16 offset,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

（1）TPM\_RC\_NV\_LOCKED 如果NV的TPMA\_NV\_WRITELOCKED属性（写锁）设置为1，则返回该错误。

（2）TPM\_RC\_ATTRIBUTES 当TPM\_NT为TPM\_NT\_COUNTER, TPM\_NT\_BITS 或TPM\_NT\_EXTEND时，TPM将会返回TPM\_RC\_ATTRIBUTES错误。

（3）TPM\_RC\_NV\_RANGE offset 和 size加起来的大小大于 nv中的数据总大小。

（4）TPM\_RC\_NV\_RANGE 当TPMA\_NV\_WRITEALL属性为1时，如果size的大小与NV的数据域大小不一致时，将会返回该错误。

（4）TPM\_RC\_VALUE offset的大小大于nv中的数据总大小，则返回该错误。

注：

1. 该命令成功执行后，会将TPMA\_NV\_WRITTEN设置为1。
2. 一旦TPMA\_NV\_WRITTEN被置位，将会一直保持，直到NV被undefined或者NV index被clear。
   * 1. Tss2\_Sys\_NV\_ ReadLock

**函数名：**Tss2\_Sys\_NV\_ ReadLock（）

**功能描述：**给当前的NV index设置读锁，使后续的读取操作失败。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_ReadLock(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

TPM\_RC\_ATTRIBUTES 如果TPMA\_NV\_READ\_STCLEAR属性没有设置，则会返回该错误。

注：

1. 设置读锁并不影响对该NV Index的写入操作。
2. 已经设置过读锁，再次调用也不会返回失败。
3. 即时定义的NV没有进行过写操作，readlock也不会返回失败。
   * 1. Tss2\_Sys\_NV\_ WriteLock

**函数名：**Tss2\_Sys\_NV\_ WriteLock（）

**功能描述：**给当前的NV index设置读锁，使后续的读取操作失败。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_WriteLock(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

TPM\_RC\_ATTRIBUTES 如果TPMA\_NV\_WRITEDEFINE或者TPMA\_NV\_WRITE\_STCLEAR属性都没有设置，则会返回该错误。

注：

1. 设置写锁并不影响对该NV Index的读操作。
2. 已经设置过写锁，再次调用也不会返回失败。
3. 即使定义的NV没有进行过写操作，writelock也不会返回失败。
   * 1. Tss2\_Sys\_NV\_ReadPublic

**函数名：**Tss2\_Sys\_NV\_ ReadPublic（）

**功能描述：**

读取指定NV index的name和public区域数据。由于public区域的数据不是敏感数据，所以读该区域不需要授权。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_ReadPublic(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_NV\_PUBLIC \*nvPublic,

TPM2B\_NAME \*nvName,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| nvIndex | TPMI\_RH\_NV\_INDEX | Nv Index |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS const \* | 授权信息（该函数不需要） |
| \*nvPublic | TPM2B\_NV\_PUBLIC\* | 输出参数（public data） |
| \*nvName | TPM2B\_NAME \* | 输出参数（NV name）  typedef union {  struct {  UINT16 size;  BYTE name[sizeof(TPMU\_NAME)]; } t;  TPM2B b;  } TPM2B\_NAME; |

注：

1. Readpublic操作不需要授权，所以cmdAuthsArray参数可以直接传NULL。
2. nvName参数需要指定name中size，以告诉TPM buffer大小。
   * 1. Tss2\_Sys\_NV\_ GlobalWriteLock

**函数名：**Tss2\_Sys\_NV\_ GlobalWriteLock（）

**功能描述：**

该函数功能是将具有TPMA\_NV\_GLOBALLOCK属性的所有NV的TPMA\_NV\_WRITELOCKED位设置为1（即将所有具有TPMA\_NV\_GLOBALLOCK属性的NV都加上写锁）。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_GlobalWriteLock(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_PROVISION authHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sysContext, | TSS2\_SYS\_CONTEXT\* | 上下文 |
| authHandle | TPMI\_RH\_PROVISION | 授权（平台或者owner）TPM\_RH\_OWNER or TPM\_RH\_PLATFORM |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS\* | 输出信息 |

注：

* 如果NV既具有TPMA\_NV\_WRITELOCKED又具有TPMA\_NV\_WRITEDEFINE属性时，Tss2\_Sys\_NV\_GlobalWriteLock命令将永久给该NV加写锁，除非该NV的TPMA\_NV\_WRITTEN被clear。
  + 1. Tss2\_Sys\_NV\_ Increment

**函数名：**Tss2\_Sys\_NV\_ Increment（）

**功能描述：**该函数功能是使TPM\_NT = TPM\_NT\_COUNTER的Nvindex的data value值加1。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_Increment(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_ATTRIBUTES TPM\_NT != TPM\_NT\_COUNTER，则返回该错误。
2. TPM\_RC\_NV\_LOCKED 如果nv具有写锁（即TPMA\_NV\_WRITELOCKED=1），则返回该错误。

注：

* 如果TPMA\_NV\_WRITTEN = 0，函数执行成功后，会将该位置1。
* 当TPMA\_NV\_WRITTEN = 0时，会将初始值初始化为 所有的TPM\_NT = TPM\_NT\_COUNTER的NV中counter value值中最大值加1。或者是曾经的最大值加1。
* 如果TPMA\_NV\_ORDERLY = 1，该字段的易失性（RAM中的）和非易失性（NV）版本之前的差异在于，当counter的值大于MAX\_ORDERLY\_COUNT时，NV需要更新。
  + 1. Tss2\_Sys\_NV\_ Extend

**函数名：**Tss2\_Sys\_NV\_ Extend（）

**功能描述：**

该函数功能是将一个值extend到NV memory中（将将要扩展的数据放在nv原来数值之后，组成新的字符串，然后根据当前NV指定的hash算法计算摘要值）。

算法nvIndex→datanew ≔ HnameAlg(nvIndex→dataold || data.buffer)

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_Extend(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_MAX\_NV\_BUFFER \*data,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_ATTRIBUTES TPM\_NT != TPM\_NT\_EXTEND，则返回该错误。
2. TPM\_RC\_NV\_LOCKED 如果nv具有写锁（即TPMA\_NV\_WRITELOCKED=1），则返回该错误。

注：

* 如果TPMA\_NV\_WRITTEN = 0，函数执行成功后，会将该位置1。
  + 1. Tss2\_Sys\_NV\_ SetBits

**函数名：**Tss2\_Sys\_NV\_ SetBits（）

**功能描述：**该函数功能用于将NV的bit位域进行set bits，该函数只是进行或运行算。

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_SetBits(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

UINT64 bits,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_ATTRIBUTES TPM\_NT != TPM\_NT\_BITS，则返回该错误。

注：

* 如果TPMA\_NV\_WRITTEN = 0，函数执行成功后，会将该位置1，无论bit位是否被set。
* 当TPM\_NT == TPM\_NT\_BITS，必须dataSize= 8。
  + 1. Tss2\_Sys\_NV\_ ChangeAuth（未验证）

**函数名：**Tss2\_Sys\_NV\_ ChangeAuth（）

**功能描述：**

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_ChangeAuth(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \* cmdAuthsArray,

TPM2B\_AUTH \*newAuth,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

* + 1. Tss2\_Sys\_NV\_ Certify（未验证）

**函数名：**Tss2\_Sys\_NV\_ Certify（）

**功能描述：**

**接口定义：**

TPM\_RC Tss2\_Sys\_NV\_Certify(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_DH\_OBJECT signHandle,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_DATA \*qualifyingData,

TPMT\_SIG\_SCHEME \*inScheme,

UINT16 size,

UINT16 offset,

TPM2B\_ATTEST \*certifyInfo,

TPMT\_SIGNATURE \*signature,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_KEY If the sign attribute is not SET
2. TPM\_RC\_NV\_UNINITIALIZED 该NV已经定义但未进行写操作。
3. TPM\_RC\_NV\_RANGE offset + size 大于NV datasize
4. TPM\_RC\_VALUE offset大于NV的datasize 或者 size大于 datasize
5. Session相关命令
   1. Session相关命令
      1. Tss2\_Sys\_StartAuthSession

**函数名：**Tss2\_Sys\_StartAuthSession（）

**功能描述：**

使用建立会话密钥（session key）的替代方法来启动一个授权会话。 然后会话密钥用于导出用于授权和加密参数的值。

这条命令启动一个授权会话，并返回session handle句柄和一个初始的nonce。

**接口定义：**

TPM\_RC Tss2\_Sys\_StartAuthSession (

TSS2\_SYS\_CONTEXT \*sysContext, //in

TPMI\_DH\_OBJECT tpmKey, //in

TPMI\_DH\_ENTITY bind, //in

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray, //in

TPM2B\_NONCE \*nonceCaller, //in

TPM2B\_ENCRYPTED\_SECRET \* encryptedSalt, //in

TPM\_SE sessionType, //in

TPMT\_SYM\_DEF \*symmetric, //in

TPMI\_ALG\_HASH authHash, //in

TPMI\_SH\_AUTH\_SESSION \*sessionHandle, //out

TPM2B\_NONCE \*nonceTPM, //out

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray //out

)

**输入参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 参数 |
| \*sysContext | TSS2\_SYS\_CONTEXT \* | 上下文 |
| tpmKey | TPMI\_DH\_OBJECT | 加密类型 |
| bind | TPMI\_DH\_ENTITY | 使用bind参数引用的实体为sessionKey生成过程贡献一个授权值。 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS \* | 授权信息 |
| \*nonceCaller | TPM2B\_NONCE\* |  |
| \*encryptedSalt | TPM2B\_ENCRYPTED\_SECRET\* | 根据tpmKey来确定该采纳数的的加密方式。如果tpmKey = TPM\_RH\_NULL，则改参数为空。 |
| sessionType | TPM\_SE | Session类型（以下三种）  TPM\_SE\_HMAC  TPM\_SE\_POLICY  TPM\_SE\_TRIAL |
| \*symmetric | TPMT\_SYM\_DEF\* | typedef struct {  TPMI\_ALG\_SYM algorithm; /\*对称算法 \*/  TPMU\_SYM\_KEY\_BITS keyBits; /\* a supported key size \*/  TPMU\_SYM\_MODE mode;  } TPMT\_SYM\_DEF; |
| authHash | TPMI\_ALG\_HASH | 指定使用的hash算法 |

**输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 描述 |
| \*sessionHandle | TPMI\_SH\_AUTH\_SESSION | 返回会话句柄 |
| \*nonceTPM | TPM2B\_NONCE | Nonce |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS | 返回授权信息 |

**返回值：**

1. TPM\_RC\_SESSION\_HANDLES TPM没有空闲的session slot时，返回该错误。
2. TPM\_RC\_CONTEXT\_GAP TPM实现了用于分配contextID 值的gap功能，如果创建session将阻止旧的已保存的上下文回收的话。
   1. PM\_RC\_HANDLE 如果tpmKey的敏感数据没有加载，将返回该错误。
3. TPM\_RC\_VALUE

A tpmKey引用RSA密钥且

（1）encryptedSalt的大小与tpmKey的公开modulus的大小不相等

（2）encryptedSalt的大小pmKey的公开modulus的大小

（3）encryptedSalt不是一个正确编码的OAEP值。

（4）解密的盐值大于由tpmKey的nameAlg产生的摘要的大小;

B tpmKey引用ECC密钥且encryptedSalt

（1）不包含一个TPMS\_ECC\_POINT

（1）不是tpmKey曲线上的一点;

1. TPM\_RC\_KEY 如果tpmkey不引用非对称密钥，TPM将返回该值。
2. TPM\_RC\_SCHEME the scheme of the key is not TPM\_ALG\_OAEP or TPM\_ALG\_NULL。
3. PM\_RC\_HANDLE bind的参数引用一个临时变量，且对象的敏感数据没有被加载。
4. TPM\_RC\_SIZE 如果nonceCaller小于16字节，或者大于算法产生的摘要值。

注：

1. 如果tpmkey和bind都是TPM\_RH\_NULL，sessionKey将被设置为空buffer。如果tpmKey不是TPM\_RH\_NULL，则在计算sessionKey时使用encryptedSalt。如果bind不为TPM\_RH\_NULL，则在sessionKey计算中使用bind的authValue。
2. Symmetric指定为一种块加密，则Symmetric.mode的值只能是TPM\_ALG\_CFB。
3. 对于所有的会话类型，这条命令会初始化sessionKey并且建立将object和session绑定。 如果session的类型是TPM\_SE\_POLICY or TPM\_SE\_TRIAL,额外的session初始化：
4. set policySession->policyDigest to a Zero Digest (the digest size for policySession->policyDigest is the size of the digest produced by authHash);
5. authorization may be given at any locality;
6. authorization may apply to any command code;
7. authorization may apply to any command parameters or handles;
8. the authorization has no time limit;
9. an authValue is not needed when the authorization is used;
10. the session is not bound;
11. the session is not an audit session; and
12. the time at which the policy session was created is recorded.
13. 如果session的类型是TPM\_SE\_TRIAL，会话将不能被用于授权，但是可以用于计算object的authPolicy
    * 1. Tss2\_Sys\_GetSessionAuditDigest(未验证)

**函数名：**Tss2\_Sys\_GetSessionAuditDigest（）

**功能描述：** 用于获取审计会话的数字签名

**接口定义：**

TPM\_RC Tss2\_Sys\_GetSessionAuditDigest(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_ENDORSEMENT privacyAdminHandle,

TPMI\_DH\_OBJECT signHandle,

TPMI\_SH\_HMAC sessionHandle,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_DATA \*qualifyingData,

TPMT\_SIG\_SCHEME \*inScheme,

TPM2B\_ATTEST \*auditInfo,

TPMT\_SIGNATURE \*signature,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1、TPM\_RC\_TYPE. 如果session不是audit session，则返回该错误。

注：

* 在会话首次用作审计会话的命令成功完成之前，会话不会成为审计会话。
* 如果此命令被审计，那么签名的审计摘要将不包括此命令的摘要，因为只有当命令成功完成时才更新审计摘要。
* 此命令不会导致审核会话关闭，并且不会重置摘要值。
* 如果sessionHandle用作此命令的审计会话，则该命令的审核方式与其他任何命令相同。
* 如果signHandle== TPM\_RH\_NULL时，scheme->scheme = TPM\_ALG\_NULL; scheme->details.any.hashAlg = TPM\_ALG\_NULL ， TPMS\_ATTEST结构体将会被返回，并且signature是一个空结构体。

1. Policy相关命令
   1. Policy相关命令
      1. Tss2\_Sys\_PolicyNV

**函数名：**Tss2\_Sys\_PolicyNV（）

**功能描述：**用于按照operation指定的方式验证NV index中的数据来跟operandB中数据的关系。

**接口定义：**

TPM\_RC Tss2\_Sys\_PolicyNV(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_RH\_NV\_AUTH authHandle,

TPMI\_RH\_NV\_INDEX nvIndex,

TPMI\_SH\_POLICY policySession,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPM2B\_OPERAND \*operandB,

UINT16 offset,

TPM\_EO operation,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**输入输出参数：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 参数类型 | 参数 |
| \*sysContext | TSS2\_SYS\_CONTEXT\* | 上下文 |
| authHandle | TPMI\_RH\_NV\_AUTH | 授权人：  TPM\_RH\_PLATFORM  TPM\_RH\_OWNER |
| nvIndex | TPMI\_RH\_NV\_INDEX | 要验证的NV的索引值 |
| policySession | TPMI\_SH\_POLICY | 被扩展的策略会话句柄 |
| \*cmdAuthsArray | TSS2\_SYS\_CMD\_AUTHS\* | 授权信息 |
| \*operandB | TPM2B\_OPERAND\* | 将要与NV数据进行check的数据。 |
| Offset | UINT16 | 从nv数据读取的偏移量 |
| operation | TPM\_EO | 操作类型：  TPM\_EO\_EQ /\* A==B \*/  TPM\_EO\_NEQ /\* A!= B \*/  TPM\_EO\_SIGNED\_GT /\* A > B signed \*/  TPM\_EO\_UNSIGNED\_GT /\* A > B unsigned \*/  TPM\_EO\_SIGNED\_LT /\* A < B signed \*/  TPM\_EO\_UNSIGNED\_LT /\* A < B unsigned \*/  TPM\_EO\_SIGNED\_GE /\* A B signed \*/  TPM\_EO\_UNSIGNED\_GE /\* A B unsigned \*/  TPM\_EO\_SIGNED\_LE /\* A B signed \*/  TPM\_EO\_UNSIGNED\_LE/\* A B unsigned \*/  TPM\_EO\_BITSET /\* All bits SET in B are SET in A. ABB \*/  TPM\_EO\_BITCLEAR /\* All bits SET in B are CLEAR in A. AB0 \*/ |
| \*rspAuthsArray | TSS2\_SYS\_RSP\_AUTHS \* | 输出授权 |

**返回值：**

1. TPM\_RC\_NV\_UNINITIALIZED TPMA\_NV\_WRITTEN != SET时，该函数返回该值。
2. TPM\_RC\_NV\_LOCKED TPMA\_NV\_READLOCKED == SET时，该函数返回错误。
3. TPM\_RC\_POLICY 如果TPM使用operandB执行算术检查，检查失败再返回该错误。
4. 如果检查成功，就会修改session的policyDigest成员。

args ≔ **H**policyAlg(operandB.buffer || offset || operation)，

policyDigestnew ≔ **H**policyAlg(policyDigestold || TPM\_CC\_PolicyNV || args || nvIndex→Name)

注:

* TPM\_NT = TPM\_NT\_COUNTER 或TPM\_NT\_BITS时，该函数传入的offset将被忽略使offset = 0。
* TPM\_RC\_NV\_RANGE offset 和size 加起来大于dataSize。
* TPM\_RC\_VALUE offset大于dataSize。
* 如果session的类型是TPM\_SE\_TRIAL，该函数将只更新policySession→policyDigest但不会进行nv data的数据验证。
  + 1. Tss2\_Sys\_PolicyNvWritten

**函数名：**Tss2\_Sys\_PolicyNvWritten（）

**功能描述：**用于按照operation指定的方式验证NV index中的数据来跟operandB中数据的关系。

**接口定义：**

TPM\_RC Tss2\_Sys\_PolicyNvWritten(

TSS2\_SYS\_CONTEXT \*sysContext,

TPMI\_SH\_POLICY policySession,

TSS2\_SYS\_CMD\_AUTHS const \*cmdAuthsArray,

TPMI\_YES\_NO writtenSet,

TSS2\_SYS\_RSP\_AUTHS \*rspAuthsArray

)

**返回值：**

1. TPM\_RC\_VALUE

当policySession→checkNVWritten == SET 且 policySession->nvWrittenState的值与writtenSet的值不同时将返回错误码TPM\_RC\_VALUE。

注：

* policySession→checkNVWritten==clear， 他将被置位，并且policySession→nvWrittenState将被设置为writtenSet.
* 如果该命令执行未返回错误，则该命令将更新policySession→policyDigest

policyDigestnew ≔ **H**policyAlg(policyDigestold || TPM\_CC\_PolicyNvWritten || writtenSet)

* 当一个索引被写入时，它具有与未写入的索引不同的授权名称。 这个在nv index的改变特性可以被用于创建一次写入的索引。