介绍 TPM 的编程

TSS/裤子基础知识

大卫·查林纳
约翰霍普金斯大学应用物理实验室

目录

4pl

准备好你的机器

假设：您正在使用 Fedora12Linux 或 UbuntuLinux 与 gcc

主安装(Fedora12Linux w/gcc)

- 玉安装裤子
- 安装 TPM 工具
- 尤姆安装裤子碎片
- 安装 GCC

Ubuntu Linux w/gcc sudo
apt-get install suters sudo apt-get
install tpm-tools sudo apt-get
install libtspi-dev sudo apt-get
install gcc

打开 TPM

去 BIOS 并确保 TPM 打开

- (如果它是，而且你不知道所有者 Auth，你可能想清除它并重新开始)。
- 不幸的是，程序从 PC 到 PC 不同

启动 tcsd(sudotcsd start)

确保您可以运行 TPM 工具(使用 tpm_getpubek)

使用 tpm_takeownership-z 取得所有权

- (-z 将 SRK 密码设置为所有零，默认的“众所周知的秘密”)
- 使用 123 为所有者 Auth 为该类

注意：如果您的机器没有在其 ACPI 表中列出的 TPM，则仍然可以让设备驱动程序使用它

在这种情况下，您必须使用：

- 苏多模探 tpm_tis 力=1 中断=0
- 苏多 TCSD 开始

4pl

注释：示例代码

- 裤子测试套件至少练习一次每个命令。
- 因此，使用每个命令的示例代码是可用的
  http://sourceforge.net/projects/trousers/files/
  下载 TSSAPI 测试套件

包括在内

```
//Basic 包括如下：

#包括<stdio.h>
#包括<string.h>

#包括<TSS/tss_error.h>
#包括<tss/platform.h>
#包括<TSS/tss_defines.h>
#包括<TSS/tss_typedef.h>
#包括<TSS/tss_structs.h>
```

#包括<TSS/TSPI H>
包括<裤子/裤子。h>

错误报告

如果裤子 api 失败，您需要将它给您的错误代码翻译成英文

幸运的是，这已经被编码成裤子了

包括<裤子/裤子。h>在你的包括

使用调试语句如下：.

```
#定义 DEBUG0
如果(DEBUG){fprintf("(行%d，%s)%s 返回 0x%08x，#定义 DBG(消息，t 结果)。 %s.n"_____line_____ _，____func___，消息，t 结果，trspi_Error_String(t
    结果)；}
```

示例使用：DBG（“创建了我的签名密钥”，结果）；

APL

序言（几乎在每个程序中）

```
主(intargc，char**argv)
{
    TSS_HCONTEXT  h 上下文=0；
    TSS_HTPM      h TPM=0；
    TSS_HKEY      结果；
    TSS_UUID BYTE h SRK=0；
    模集(wks，0，20)；h SRKPolicy=0；
                  SRK_UUID=TSS_UUID_SRK；wks[20]；/放置众所周知的秘密/设置 wks 到众所周知的秘密 20 字节的所有零

                  //选择您正在讨论的 TPM，在这种情况下，系统 TPM(您连接到“NULL”)结果=Tspi_Context_Create(&h 上下文)；
                      结果=Tspi_Context_Connect(h 上下文，NULL)；
                                                                    DBG(“Create a Context\n”，result)；
//拿 TPM 手柄                                                       DBG(“连接到 TPM\n”，结果)；
  结果=Tspi_Context_GetTpmObject(h 上下文，&hTP M)；
                                                                     DBG(“获取 TPM Handle\n”，结果)/
找到 SRK 手柄
  结果=Tspi_Context_LoadKeyByUUID(h 上下文、TSS_PS_TYPE_SYSTEM、SRK_UUID 和 hSRK)；
                                                                    DBG(“Tspi_Context_Connect\n”，结果)；
/制定 SRK 政策
  结果=Tspi_GetPolicyObject(hSRK，TSS_POLICY_USAGE，&hSRKPolicy)；
                                                                    DBG(“Get TPM Policy\n”，result；)；
//然后我们将 SRK 策略设置为众所周知的秘密 result=Tspi_Policy_SetSecret(hSRKPolicy，TSS_SECRET_MODE_SHA1,20，wks)；//注意：TSS_SECRET_MODE_SHA1 说“不要
  散列这个。 只需按原样使用 20 个字节。
                                                                    DBG(“T spi_Policy_Set_Secret\n”，结果)；
```


APL

清理（在每个程序的末尾）

```
/*清理*/
    spi_Context_Close(您创建的 h 对象);
    spi_Context_FreeMemory(h 上下文，空);
     //这释放了自动为您分配的内存 Tspi_Context_Close(hContext);
  返回 0;
}
gcc file-o file.exe-Itspi-Wall
```

关于内存处理的评论

- **如果函数调用 BYTE*，那么 TCS 将为您分配内存的可能性很大。**
  如果你想找的话，说明书应该告诉你。

- **这意味着你需要**
- **将变量定义为.**

字节 ^变量;

使用变量作为和变量；

这样 TCS 就可以将内存分配给未分配的指针

- **注意：如果你做一些愚蠢的事情，比如**

字节　　可变[256]；

传球　　变量；

它会做不可预测的事情！！

例如

原型：

TSS_RESULT Tspi_Hash_Sign

TSS_HHASH h Hash，　　//进去
tss_hkey　　h Key，　　//进去
uint32*　　pul 签名长度，　　//出去
字节　　PRGB 签名　　//出去**

);

守则：

uint32　　签字长度；
*字节　　*rgb 签字结果；*
tss_result　　h Hash；h Key；
tss_hhash 是史基

结果=Tspi_Hash_Sign(hHash，h 键，&签名长度，&rbg 签名)；

中间该怎么办.

创建对象

玩他们的属性（获取属性，设置属性）

- 您可以使用的属性在具有该对象函数的部分的规范中列出

创建一个与对象关联的 Policy 对象

将 Policy 对象与另一个对象关联

-硅中的不稳定物体(Key_CreateKey 等)。)

登记一把钥匙

使用一个对象

- 签名/签名/签名/未绑定/验证签名/用键报价
- 读/写 NVRAM
- 读/延伸/重置 PCR

授权

- **TPM 要求每次使用密钥时都使用密钥的授权。**
- **如果用户每次使用密钥都必须输入密码一他就不会使用密钥**
- **TSS 解决方案：**

  告诉 TCS 上下文对象的密码——一次

  -以后每一次（在那个程序中），它都会记住并使用它

否则，创建不需要授权的对象

中间授权

让 TSS 知道特定对象的授权，如密钥或 TPM：

定义策略对象句柄

TSS_HPOLICY 我的政策处理；

- **将策略句柄与策略对象关联**
  - 获取现有策略(例如。 对于 TPM)结果=Tspi_GetPolicyObject(hTPM，TSS_POLICY_USAGE，和 hTPMPolicy)；
  - 创建策略（稍后将其与对象关联）

  结果=Tspi_Context_CreateObj 等(h 上下文、TSS_OBJECT_TYPE_POLICY、TSS_POLICY_USAGE 和 h 新政策)；
- **将授权值填写到 Policy 对象中.**

  Tspi 策略设置秘密(h 新策略，TSS 的 ECRET 模式 PLAIN，密码长度，*新密码)；

将 Policy Object 与适当的对象关联（如果您没有从一个对象获得一个现有的策略）

Tspi_Policy_AssignToObject(h 新策略，h 对象)；

示例代码

```
//获取 TPM 的策略对象
TSS_HPOLICY hTP MPolicy；〒Esult=Tspi_GetPolicyObject(hTPM，TSS_POLICY_USAGE 和 hTPMPolicy)；
                                        DBG(“Tspi_GetPolicyObject TPM 策略”，结果)；

/*然后，我们将默认所有者授权设置为其秘密*/result=Tspi_Policy_SetSecret(hOldTPMPolicy，
```


4pl

TSS_SECRET_MODE_PLAIN，3，“123”)；/注：3=strlen（“123”）DB G(“T spi_Policy_Set_Secret”，结果)；

/*创建新策略并将新密码放入其中*/

结果=Tspi_Context_CreateObject(h 上下文、TS S_OBJECT_TYPE_POLICY、TSS_POLICY_USAGE 和 h 新政策)；
DBG(“T spi_Context_CreateObj 等策略对象”，结果)；

结果=Tspi_Policy_SetSecret(h 新策略，TS S_SECRET_MODE_PLAIN，20，*新密码)；
d BG(“T spi_Policy_SetSecret”，结果)；

*将密码更改为新策略中的密码

结果=Tspi_ChangeAuth(hTPM，Ox OOOOOO O0，h 新策略)；/(0x00000000 是 TPM 的父级)。 DBG(“Change Auth of Tp M”，result)；

Keys

钥匙的类型

- 储存（密封）.
  - 锁定密码+PCR，并在创建时记录 PC R 值

- 具有约束力
  - 只锁定密码(但密钥可以锁定到 PCR)

艾克

- 都限制在他们能做什么，目前只用 Tspi_Key_CollateIdentityKey 命令创建.

  只能是 2048RSA 的不可移动密钥.

- 签名
  - 可以做任何 AIK 可以做的，再加上更多。 可迁移或不可迁移.. 可以是 1024 或者 2048 按键.

    一 4sig 模式是可能的。 其中之一，只 TSS_SS_RSASSAPKCS11V15_INFO 标志结构，故不可欺骗.
- 遗产
  - 既能装订又能签字.. 危险，但用于向后兼容.

- （在未来，由特征定义）

APL

创建密钥

创建对象（按类型）

填写你知道的/想要的键看起来像授权+PCR 锁定

钥匙的尺寸

父键的可迁移/不可迁移句柄

加载父级(如果不是 SRK)

请 TPM 填空

Tspi_Key_CreateKey（除非身份密钥）

Tspi_TPM_CollateldentityRequest（如果是身份密钥）

把钥匙放下

注册密钥(由 UUID)

提取加密的密钥 BLOB 并将其存储在文件中

提取公钥并存储在文件中

APL

代码示例：创建绑定密钥

```
定义 BACKUP_KEY_UUID{0，0，0，0，0，0，{0，0，0，0，2，10}
tss_hkey                    hBackup_Key；
tss_uuid                    my_uuid=backup_key_uuid；
tss_hpolicy          hBackup_Policy；
tss_flag             伊尼特旗；
字节                 *pub Key；
uint32               酒吧钥匙大小；
文件                 *失败；
/*为新密钥创建策略。 我会把密码设置为“123”*/
        结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_POLICY，0，&hBackup_Policy)；
          DBG(“Tspi_Context_CreateObject 策略\n”，结果)；
        Tspi_Policy_SetSecret(hBackup_Policy、TSS_SECRET_MODE_PLAIN、3、“123”);          //SECRET_MODE_PLAIN 意味着它需要在使用前进行散列
          DBG（“设置秘密”，结果）；
/*实例化一个密钥对象，该对象是需要授权的类型为“BIND”的 2048 位 RSA 密钥。 */
        initFlags=TSS_KEY_TYPE_BIND|TSS_KEY_SIZE_2048|TSS_KEY_AUTHORIZATION|TSS_KEY_NOT_MIGRATABLE；//Section2.3.2.2has choices result=T
        spi_Context_CreateObject(hContext，TSS_OBJECT_TYPE_RSAKEY，initFlags，&hBackup_Key)；DBG(“Tspi_Context_CreateObject Key\n”，result)；
*指导政策
        结果=T spi_Policy_AssignT o 对象(hBackup_Policy，hBackup_Key)；DB G(“T spi_Policy_AssignT oObject\n”，结果)；/不能分配策略，直到您有句柄/*创建并注册它*/
        结果=Tspi_Key_CreateKey(hBackup_Key，hSRK，NULL)；    DBG(“Tspi_Key_CreateKey\n”，结果)；              //让 TPM 填空
        结果=Tspi_Key_LoadKey(hBackup_Key，hSR K)；
        结果=TSPI 上下文寄存器键(hContext，hBackup 键，TSSPS 类型系统，MYUID，TSSPS 类型系统，SRKUID)；如果(结果！=TSS_SUCCESS){DB G(“Tspi_Context_RegisterKey\n”，结果)；
          返回 1；}

/*现在密钥已经注册，我还想将密钥的公共部分存储在一个文件中，以供分发*/这是分两部分完成的：1)获取公钥，2)将其放入 Backup.pub result=Tspi_Key_GetPubKey(hBackup_Key，&pubKeySize，
&pubkey)；
          如果(结果！=TSS_SUCCESS){DBG(“Tspi_Key_GetPubKey\n，结果”)；返回 1；}print f(“Erro^%s”，(char*)Trspi_Error_String（结果）；
//        2)把它保存在一个文件中。 文件名为“Backup.pub”
        fout=fopen(“backup.pub”，“w”)；

          fclose(fout)；
        写(Fileno(Fout)，酒吧键，酒吧键大小)；
```


APL

创建一个签名键，注册它并获得它的公共部分

定义 TSS_UUID_SIGN{0，0，0，0，0，0，{0，0，0，0，2，0}/用户签名密钥 1
UINT32 酒吧钥匙长度；
BYTE*pub Key；

```
//我们要创建一个标志

//这里，我确定密钥将是一个 2048 位的签名密钥，不可迁移，没有授权。
```

```
=TSS_KEY_TYPE_SIGNING|||TSS_KEY_NOT_MIGRATABLE；/创建关键对象

结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_RSAKEY，init 标志，&hSigning_Key)；

DBG(“Tspi_Context_CreateObjectSigningKey”，result)；
```

```
//现在我终于创建了密钥，SRK 是它的父级。打印 f(“创建密钥.这可能需要一些时间\n”)；结果=Tspi_Key_CreateKey(hSigning_Key，hSRKey，0)；DBG（“创建密钥”，结果）；

//一旦创建，我注册密钥 BLOB，以便我可以在稍后检索它 result=Tspi_Context_RegisterKey(hContext，hSigning_Key，TSS_PS_TYPE_SYSTEM、SIGNING_UUID、
TSS_PS_TYPE_SYSTEM、SRK_UUID)；DBG（“注册密钥”，结果）；
```

/*现在密钥已经注册了，我还想将密钥的公共部分存储在一个文件中以供分发*//*这分为两部分：1)加载密钥并读出公钥并将其输入 pubKey*/

```
result=Tspi_Key_LoadKey(hSigning_Key，hSRKey)；
DBG(“Load Key”，result)；
结果=Tspi_Key_GetPubKey(hSignking_Key，&酒吧钥匙长度，&酒吧钥匙)
```


4pl

创建 AIK

需要 Owner_auth

制定 TPM 政策

-保密 Owner_auth

- **获取 SRK 句柄（从序言）**
- **创建对象(AIK 类型的键)**
- **填写你知道的内容（关键尺寸等）。)**
- **Tspi_CollateIdentityRequest**

  隐式使用 TPMauth

  需要 CA 公共密钥、EK 公共密钥等。 通常是假的
- **注册它，以便您可以在稍后通过 UUID 找到它**

样本代码

前往：http://www.privacyca.com/code.html 用于创建 AIK 的示例代码

由 UUID 加载密钥

通过 UUID 获取密钥

，加载键

UUID 加载密钥在 Trou SerS 中不起作用，除非父密钥被 No_Auth。 注意 SRK 使用的“众所周知的秘密”不是一个 no_auth 的密钥。

然而，这是您获得 SRK 句柄的方式

//拿 SRK 手柄

结果=Tspi_Context_LoadKeyByUUID(h 上下文、TSS_PS_TYPE_SYSTEM、SRK_UUID 和 hSR K)；

如果(结果！=Tspi_Context_LoadKeyByUUIDTSS_SUCCESS){DBG(“Tspi_Context_Connect\n^，结果)；返回 1；}

样本代码

TSS_HKEY hBind_Key=0；

Tspi_Context_GetKeyByUUID(h 上下文、TSS_PS_TYPE_SYSTEM、BACKUP_KEY_UUID 和 hBind_Key)；
Tspi_Key_LoadKey(hBind_Key、hSRK)；

//克莱纳普
Tspi_Context_CloseObject(h 上下文，hBind_Key)；

拿一把公钥，给它手柄

- 使用 Get 属性（第 4.3.4.18.4 节）
  或者
- 使用 Tspi 键获取酒吧键

将公钥保存到文件中

样本代码

```
uint32          酒吧钥匙大小；
字节            *pubkey；/(不要使用 pubkey[284]；)
file*fout；
//获取公钥(可以使用此或获取 PubKey)结果=Tspi_GetAttribData(hSigning_Key，
                    tss_tspattrib_key_blob，
    TSS_TSPATTRIB_KEYBLOB_PUBLIC_KEY，
                    酒吧钥匙大小，酒吧钥匙)；
     DBG（“从关键对象获取公钥”，结果）；

//2)将公钥保存在文件中。 文件名为“Signing.pub”fout=fopen(“Signing.pub”，“w”)；
        写(Fileno(Fout)，酒吧键，284)；/或写(Fileno(Fout)，酒吧键，酒吧键大小)；fclose(Fout)；
```

一数据对象绑定数据.

加载绑定密钥（只有公钥是必要的）

- 创建一个数据对象
- 填写清晰的文本和“绑定”（加密）数据
- 读出加密数据.

数据对象

加密数据

样本代码

UINT32ul 数据长度；BYTE*rbg 绑定数据；

//检索公钥

fin=fopen(“Bind.pub”，“r”)；

阅读(Fileno(Fin)，新的酒吧键，284)；

fclose（鳍）；

//创建一个关键对象

结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_RSAKEY，init 标志，&hBind_Key)；DBG(“Tspi_Context_CreateObject 绑定密钥”，结果)；

//用从文件中读取的公钥输入密钥对象 result=Tspi_SetAttribData(hBind_Key，TSS_TSPATTRIB_KEY_BLOB，TSS_TSPATTRIB_KEYBLOB_PUBLIC_KEY，284，新的 PubKey)；

DBG（“将公钥设置为新密钥对象”，结果）；

//在数据中读取要加密的鳍=fopen(“AES.key”，“r”)；

读取(Fileno(FIN)，Enc 数据，7)；

fclose（鳍）；

//创建一个数据对象，用清晰的文本填充它，然后绑定它。

结果=Tspi_Context_CreateObject(h Context，TSS_OBJECT_TYPE_ENCDATA，TSS_ENCDATA_BIND，&h EncData)；DBG（“创建数据对象”，结果）；结果

=Tspi_Context_CreateObjectTspi_Data_Bind(h EncData，hBind_Key，7，EncData)；DBG（“绑定数据”，结果）；

//从数据对象结果=Tspi_GetAttribData 中获取加密数据(h、Enc 数据、TSS_TSPATTRIB_ENCDATA_BLOB，TSS_TSPATTRIB_ENCDATABLOB_BLOB，和 ul 数据长度、和 RGB 绑定数据)；

DBG（“获取加密数据”，结果）；

//将加密数据写入一个名为 Bound.data fout=fopen 的文件(“Bound.data”，“w”)；

write(fileno(fout)，rgbBoundData，ulDataLength)；fclose(fout)；

无法绑定的数据

在 TPM 中加载私钥

APL

创建绑定数据对象.

从文件读取加密数据到数据对象

将数据解绑成变量

数据对象

示例代码

```
TSS_HENCDATA h 数据；
uint32          安·伦=256；
字节            加密数据[256]；
字节            *RGB 数据未绑定；
uint32          ul 数据长度；
```

```
//从文件中读取加密数据
    鳍=开口(“绑定”。 数据“，”r“)；
      读取(Fileno(FIN)、加密数据、ul 数据长度)；
```


4pl

```
        fclose（鳍）；


//创建一个新的数据对象
        结果=Tspi_Context_CreateObject(h 上下文、TSS_OBJECT_TYPE_ENCDATA、TSS_ENCDATA_BIND 和 h 数据)；    DBG(“Create Data object”，result)；
//将加密的数据写入新的数据对象
        结果=Tspi_SetAttribData(h 数据，TSS_TSPATTRIB_ENCDATA_BLOB，TSS_TSPATTRIB_ENCDATABLOB_BLOB，，加密数据)；
                                                                                                    DBG（“设置加密数据”，结果）；
//从标准 UUID 获取无绑定私钥句柄
        Tspi_Context_GetKeyByUUID(hContext，TSS_PS_TYPE_SYSTEM，BIND_UUID 和 hUnBind_Key)；                DBG(“UUID 获取密钥”，结果)；
//使用其句柄将私钥加载到 TPM 中
        Tspi_Key_LoadKey(hRecovered_UnBind_Key，hSRKey)；                                          DBG(“Load Key”，result)；
```

```
//使用私钥将数据解密到变量 RGB 数据解绑结果=Tspi_Data_Unbind(h 新 Enc 数据，hRecovered_UnBind_Key，&ul 数据长度，&RGB 数据解绑)；
          DBG(“Unbind”，结果)；
```

密封数据

- 两种方式：.
  一创建一个绑定密钥“密封”到 PCR
  一创建密封到 PCR 的数据

- 数据密封至 PCR：.
  一创建 PCR 对象
  释放所需 PC R 值中的一文件

- 为 SEAL 创建数据对象
- 为数据对象编写清晰的文本
- 加载存储密钥
- 印章数据
- 读出加密数据

代码示例

结果=Tspi_Context_CreateObject(h 上下文，TS S_OBJE C T_TYPEPCRS，0，&hPcrs);　　DBG(“创建 PCR 对象”，结果);

结果=Tspi_Context_CreateObjectTspi_TPM_PcrRead(hTPM，8，&ulPcrLen，&RGBP CR 值)；结果

=Tspi_Context_CreateObjectTspi_PcrComposite_SetPcrValue(hPcr，8，20，RGBP CR 值)；结果

=Tspi_Context_CreateObjectTspi_TPM_PcrRead(hTPM，9，&ulPcrLen，&RGBP CR 值)；结果

=Tspi_Context_CreateObjectTspi_PcrComposite_SetPcrValue(hPcr，9，20，RGBP CR 值)；/创建加密数据对象。

//数据对象用于密封操作。 结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJE CT_ENCDATA、TSS_ENCDATA_SEAL 和 hEnc 数据)；

字符类型通行证[12]=“我的密码”；

结果=Tspi_Data_Seal(hEncData，hSRKey，strlen(TypePass)，TypePass，hPcrs)；

DBG(“读取 PC R8 的 PC R 值”，结果)；

DBG(“设定 PCR8 的当前值进行密封”，结果)；

DBG(“读取 PC R9 的 PC R 值”，结果)；

DBG(“设定 PCR9 的当前值进行密封”，结果)；

DBG（“创建一个数据对象来密封东西”，结果）；

DBG（“向数据对象分配策略”，结果）；

DBG（“用数据对象密封”，结果）；

解封数据.

- 创建数据对象（盖章类型）.
- 读取加密数据
- 将加密数据写入数据对象
- 加载键
- 打开
- 读出纯文字.

代码示例

```
uint32          长度；
字节            *外弦；
字节            加密数据[312]；
模集（加密数据，0，312）；
//读取密封数据鳍=fopen(“owner_auth.pass”，“r”)；
        读取(Fileno(FIN)，加密数据，312)；fclose(FIN)；

    结果=Tspi_SetAttribData(h 检索数据，
                            tss_tspattrib_encdata_blob，
                            TSS_TSPATTRIB_ENCDATABLOB_BLOB，
                            312,
                            加密数据)；
        DBG（“将数据对象的加密数据设置为刚刚读取的数据”，结果）；

结果=Tspi_Data_Unseal(h 检索数据，hSRKey，&outlength，&outstring)；
        DBG（“打开数据”，结果）；
```

用标牌钥匙签字

加载一个签名键

创建 Hash 对象并填充.

3 用签名密钥提取签名签署 Hash 对象并保存到文件中

样本代码

//从标准 UUID result=Tspi_Context_GetKeyByUUID(hContext，TSS_PS_TYPE_SYSTEM，SIGNING_UUID，&hSigning_Key)；DBG 获取签名密钥句柄(“通过 UUID 获取密钥”，结果)；

//使用其句柄 result=Tspi_Key_LoadKey(hSigning_Key，hSRKey)；。将私钥加载到 TPM 中　　DBG(“Load Key”，result)；

//创建一个 Hash 对象，以便有一些东西要签名，因此我们创建一个通用的 Hash 对象//结果=Tspi_Context_CreateObject(hContext、TSS_OBJECT_TYPE_HASH、TSS_HASH_SHA1 和 hHashToSign)；DB G(“创建 Hash 对象”，结果)；

APL

```
//在文件中读取哈希
        发布键长度=文件长度(“file.dat”)；
        fin=fopen(“file.dat”，“r”)；
          阅读(Fileno(FIN)，pPub 键，酒吧键长度)；
        fclose（鳍）；

//使用 SHA1/哈希数据

        结果=Tspi_Hash_UpdateHashValue(hHashtoSign，pubKeyLength，pPubKey)；          DBG(“公钥中的 Hash”，结果)；
//在结果哈希对象 result=Tspi_Hash_Sign(hHashToSign，hSigning_Key，&ulSignatureLength，&rgbSignature)；。签名      DBG（“签名”，结果）；

//将结果签名写入名为 Signature.dat 的文件

        fout=fopen(“signature.dat”，“w”)；write(fileno(fout)，rgbSignature，ulSignatureLength)；
        fclose(fout)；
```

确认签名

重新创建签名结束的散列

将公钥加载到密钥对象中

4pl

- *在签名中阅读*
- *运行验证签名*

APL

样本代码

```
//创建一个 Hash 对象，以便有一些东西来比较签名
    //创建通用 Hash 对象/
    结果=Tspi Context Create Object(h Context，TSS Object TYPE HASH，TSS HASH SHA1，&h Hash to Sign)；DBG(“Create Hash Object”，
    result)；

    pub Key Length=file length(“file·dat”)； fin=fopen(“file.dat”，“r”)；
      阅读(Fileno(FIN)，pPub 键，酒吧键长度)；fclose(FIN)；

    //使用 SHA1/哈希数据

    结果=Tspi_Hash_UpdateHashValue(hHashtoSign，pubKeyLength，pPubKey)；          DBG(“公钥中的 Hash”，结果)；

//我们将创建一个验证密钥 fin=fopen(“sign.pub”，“r”)；
      阅读(Fileno(FIN)，酒吧验证键，284)；fclose(FIN)；

    |=|=|TSS_KEY_NOT_MIGRATABLE；结果=Tspi_Context_CreateObject(h 上下文、TSS_OBJECT_TYPE_RSAKEY、init 标志和
    hVerify_Key)；
    DBG(“Tspi_Context_CreateObject Verify_Key”，结果)；                              —
    result=Tspi_SetAttribData(hVerify_Key，TSS_TSPATTRIB_KEY_BLOB，TSS_TSPATTRIB_KEYBLOB_PUBLIC_KEY，酒吧标志键
    长，酒吧验证键)；
      DBG(“Verify_Key 中设置酒吧键”，结果)；

//阅读签名并核实
    fin=fopen(“signature·dat”，“r”)；
      阅读(Fileno(FIN)，签名，256)；fclose(FIN)；

    result=Tspi_Hash_VerifySignature(hHashToSign，hVerify_Key，256，Signature)；      DBG（“验证”，结果）；
```

NVRAM

在特定的索引、特定的大小上创造空间

需要 TPM 所有者授权来定义或销毁-GetTPM 策略

在 TPM owner_auth 中填写 TPM 政策

创建 NVRAM 对象

设置特定数据（大小、索引、授权）

在指定的索引中定义空间

注释：有索引开销（每个索引大约 93 字节），因此通常不能生成无限数量的索引。 然而，您可以将多个东西放在一个特定的索引中，使用偏移量来获取它们。

示例代码(只运行一次！)

```
TSS_HNVSTORE hNVStore;
/*创建 NVRAM 对象*/
    结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_NV，0，&hNVStore);
```


4pl

如果(结果！=TSS_SUCCESS) {DBG("Tspi_Context_CreateObject：%x\n"，结果)； 返回 1；}

*其任意索引为 0x00011101(取 00-FF 和 00011600)。 */

结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_INDEX，0，0x00011101)；

如果(结果！=TSS_SUCCESS) {DBG("Tspi_SetAttribUint32 索引%x\n"，结果)；返回 1；}

/*设置其属性。 首先，它只能由所有者*/编写

结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_PERMISSIONS，0，TPM_NV_PER_OWNERWRITE)；如果(结果！=TSS_SUCCESS) {DBG("Tspi_SetAttribUint32auth%x\n"，结果)；返回 1；}

/*接下来，它保存 40 字节的数据*/

结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_DATASIZE，0，40)；

如果(结果！=TSS_SUCCESS) {DBG("Tspi_SetAttribUint32 大小%x\n"，结果)；返回 1；}

/*为了实例化或写入 NVRAM 中的 NVRAM 位置，需要 owner_auth。 在 NVRAM 的情况下，owner_auth 来自 TPM 的策略对象。 我们会把它放在这里。 */

/*首先，我们得到一个 TPM 策略对象*/

结果=Tspi_GetPolicyObject(hTPM、TSS_POLICY_USAGE 和 hTPMPolicy)；

如果(结果！=TSS_SUCCESS){DBG("Tspi_GetPolicyObject：%x\n"，结果)；返回 1；}

/*然后我们将所有者授权设置为其秘密*/

结果=Tspi_Policy_SetSecret(h TPMPolicy，TSS_SECRET_MODE_PLAIN，3，"123")；

如果(结果！=TSS_SUCCESS){DBG("Tspi_Policy_SetSecret：%x\n"，结果)；返回 1；}

/*创建 NVRAM 空间*/结果=Tspi_NV_DefineSpace(hNV Store，0，0)；

如果(结果！=TSS_SUCCESS) {DBG("Tspi_NV_DefineSpace：%x\n"，结果)；返回 1；}

NVRAM

写信给 NVRAM

创建 NVRAM 对象

设定政策秘密

在 NVRAM 对象中设置 TPM 策略秘密

写入数据

评论：虽然 TPM 知道 NVRAM 索引的密码是 TPM 所有者的密码，但 TrouSerS 无法知道这一点。 因此，您必须通过创建策略秘密、将其填充到 TPM 所有者的授权中、然后将其与 NVRAM 对象关联来告诉 TrouSerS。

示例代码(写入 NVRAM)

```
tss_hnvstore        h NVStore;
tss_hpolicy         h 新政策;
查尔                数据存储[19]=”这是一些数据。 “

/*创建 NVRAM 对象*/
    结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_NV，0，&hNVStore);
    如果(结果！=TSS_SUCCESS) {DBG(“Tspi_Context_CreateObject：%x\n”，结果);        返回 1；}

*其任意索引为 0x00011101(取 00-FF 和 00011600)。 */
```

结果=Tspi_SetAttribUint32(hNV Store，TSS_TSPATTRIB_NV_INDEX，0，0x0x00011101)；

如果(结果！=TSS_SUCCESS) {DBG(“Tspi_SetAttribUint32 索引%x\n”，结果)；返回 1；}

/*设置其属性。 首先，它只能由所有者*/编写

结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_PERMISSIONS，0，TPM_NV_PER_OWNERWRITE)；如果(结果！=TSS_SUCCESS) {DBG(“Tspi_SetAttribUint32auth%x\n”，结果)；返回 1；}

/*接下来，它保存 40 字节的数据*/

结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_DATASIZE，0，40)；

如果(结果！=TSS_SUCCESS) {DBG(“Tspi_SetAttribUint32 大小%x\n”，结果)；返回 1；}

/*使用所有者 Auth*/为 NVRAM 对象设置策略

结果=Tspi_Context_CreateObject(h 上下文、TSS_OBJECT_TYPE_POLICY、TSS_POLICY_USAGE 和 h 新政策)；结果=Tspi_Policy_SetSecret(h 新政策，TSS_SECRET_MODE_PLAIN，3，“123”)；

结果=Tspi_Policy_AssignToObject(hNewPolicy，hNVStore)；

/*写入 NVRAM 空间*/

结果=Tspi_NV_WriteValue(hNVStore，0，18，数据存储)；

如果(结果！=TSS_SUCCESS){DBG(“T spi_NV_WriteV ALUE：%x\n”，结果)；返回 1； }

NVRAM

- **从 NVRAM 读取**

  创建 NVRAM 对象

  如有需要，设置政策秘密

  读取数据

- **快于解封，因为它不需要私钥操作.**

- **高开销（约 93 字节）限制可以使用的 NVRAM 索引的数量**

  如果使用相同的授权，则可以重用相同的索引。

示例代码(从 NVRAM 读取)

```
TSS_HNVSTORE hNVStore;
查尔              数据存储[19]={0};

/*创建 NVRAM 对象*/
    结果=Tspi_Context_CreateObject(h 上下文，TSS_OBJECT_TYPE_NV，0，&hNVStore);
    如果(结果！=TSS_SUCCESS) {DBG("Tspi_Context_CreateObject：%x\n"，结果)；返回 1；}
```


APL

```
"接下来，它的任意索引将是 0x00011101(00-FF 和 00011600)。 */
    结果=Tspi_SetAttribUint32(hNV Store，TSS_TSPATTRIB_NV_INDEX，0，0x0x00011101)；
    如果(结果！=TSS_SUCCESS) {DBG("Tspi_SetAttribUint32 索引%x\n"，结果)；返回 1；}

/*设置其属性。 首先，它只能由所有者*/编写

    结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_PERMISSIONS，0，TPM_NV_PER_OWNERWRITE)；如果(结果！
    =TSS_SUCCESS)          {DBG("Tspi_SetAttribUint32auth%x\n"，结果)；返回 1；}

/*接下来，它保存 40 字节的数据*/

    结果=Tspi_SetAttribUint32(hNVStore，TSS_TSPATTRIB_NV_DATASIZE，0，40)；
    如果(结果！=TSS_SUCCESS){DBG("Tspi_SetAttribUint32 大小%x\n"，结果)；返回 1；}

 /*不需要授权从这个 NVRAM 读取它的创建方式。 /
/*从 NVRAM 空间*/结果=Tspi_NV_ReadValue(hNV Store，0，18，&数据存储[0])；
    如果(结果！=TSS_SUCCESS){DBG("Tspi_NV_ReadValue：%x\n"，结果)；返回 1；}
```

PCR 对象

- **操作 PCR**
  相反
  变化（扩展）
  重置(仅 PCR16 和 23)

- **分配 PCR 进行授权**
  钥匙
  数据（密封）

- **引用 PCR（认证）**

- **检查证明**

创建 PCR 对象，读取 PCR.

/*创建一个 PC R 合成物，其当前 PC R 值为 17 和 18。 */
/*创建 PCR 复合对象。 我使用 TSS_PCR_INFO_SHORT，因为我的 PCR>15*/

字节　　*数字值 17，*数字值 18；

结果=Tspi_Context_CreateObject(h 上下文、TSS_OBJECT_TYPE_PCRS、TSS_PCR_STRUCT_INFO_SHORT 和 hPcrs)；
DBG(“Create Object PCRs”，result)；

/*读取 PCR 指数 17 和 18，并在对象*/中设置它们的值

结果=Tspi_TPM_PcrRead(hTPM，17，&PCR 长度，&消化值 17)；　DBG(“PCR Read17”，结果)；
结果=Tspi_PcrComposite_SetPcrValue(hPcrs，17，PCR 长度，消化值 17)；　DBG(“Set PC R 值 17”，结果)；
结果=Tspi_TPM_PcrRead(hTPM，18，&PCR 长度，&消化值 18)；　DBG(“PC R Read18”，结果)；
结果=Tspi_PcrComposite_SetPcrValue(hPcrs，18，PCR 长度，消化值 18)；　DBG(“Set PC R 值 18”，结果)；

扩展一个 PCR 值

字节　　输入=“你好世界”
字节　　*Final_PCR_Value；
/把数值延长

result=Tspi_TPM_PcrExtend(hTPM，16，sizeof(myinput)，(BYTE*)输入、空、PCR_result_length 和 Final_PCR_Value)；

打印 f(“随后，PCR 编号 16 具有当前值%s\n”，F inal_PCR_Value)；

证明

- 决定你想要证明什么 PCR
- 决定你想用什么 AIK 键
- 加载 AIK
- 创建一个 PCR 对象
- 将正确的 PCR 指标放入对象中
- 将随机数设置为验证结构
- 引语

样品代码的报价和验证报价

前往：
http://www.privacyca.com/code.html
对于示例代码，既引用也验证引用

阅读日志文件（注意：使用最新的裤子）

```
uint32                  ULPC R 指数=9；
uint32                  ul 开始编号=O；
uint32                  UL 事件编号=15；
TSS_PCR_EVENT*prgbPCR 事件；char事件空白[256]；
int                     一

Tspi_TPM_GetEvents(hTPM，pcrlndex，ul 开始数，(UINT32*)和 pcr 数，&prgbPCR
                        事件)；

对于(i=0；i\textlesspcrnumber；++i){
    memset(事件 blank，0，256)；memcpy(事件 blank，prgbPcr 事件[i]。rgb 事件，
    prgbPcr 事件[i]。ul 事件日志
            )；
    打印 f(“事件%d，是%s\n”，I，事件空白)；}
```

RNG

获取 TPM 句柄（从序言）

问它一些随机字节

存储和打印随机字节

APL

样本代码

```
查尔              *随机字节；
文件              *退出；
TSS_RESULT 结果；

随机 Bytes 出=Atoi(argv[l])；
如果(随机字节=(char*)malloc(num 随机字节输出)==NULL)
```

```
/*向 TPM 询问 20 个字节的随机数，并将其填充到随机 BytesvariablT                    e*/
    spi_TPM_GetRandom(hTPM，numRandomBytesOut，random)；。中
//打印出来供用户查看
/*（我=0；我<数字随机；++一）
    (fprint(“%c02h”，随机[i])；
    }
    指纹(“\n”)；
```