

**课 程 实 验 报 告**

**课程名称： 可信计算实验**

**专业班级： 信息安全201805**

**学 号： U201810398**

**姓 名： 吴 锦 添**

**指导教师： 代 炜 琦**

**报告日期： 2021年 7月 2 日**

**网络空间安全学院**

## 实验目的

* 本实验的目的是让学生将从书本中学到的可信计算相关知识应用到实践中。在linux中使用tmpm模拟器，通过TSS软件栈调用相关硬件来完成远程证明、密钥迁移、密钥结构、数据密封等相关功能，了解TPM的安全性，学会调用TSS的各种接口来完成应用程序。
* 本实验的任务主要是在随文档提供的代码的基础下，填补代码中缺失的部分，这个工作主要在秘钥迁移和秘钥结构相关功能代码中，还有根据功能需要，补全数据密封功能所需要的代码文件。

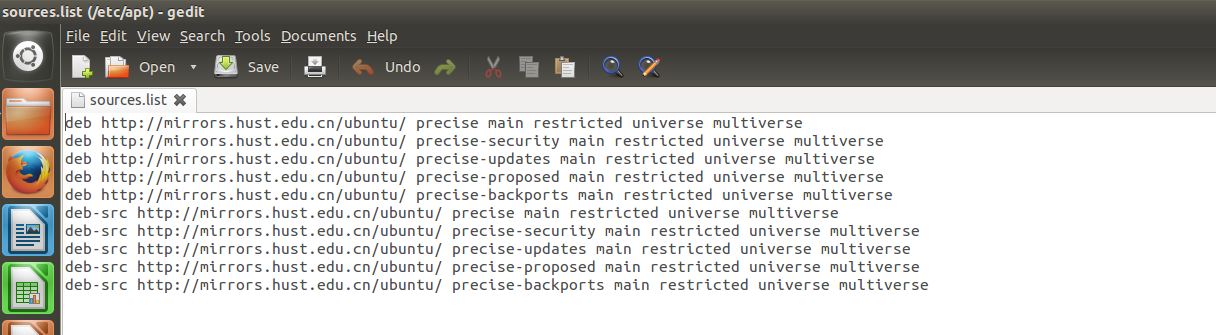
## 实验环境

* Seed Ubuntu 12.04 LTS 32位的VMware虚拟机（下载链接：https://seedsecuritylabs.org/lab\_env.html）
* TPM Emulator
* Trousers

# 实验过程

### 编译

在没有物理TPM的机器上，可以使用TPM Emulator完成TPM的相关实验。因为某些国外网站没法访问导致上述命令无法执行，首先需要换源。



在换完源而后，需要解压并编译安装TPM Emulator，相关的指令如下：

tar xvzf tpm-emulator.tar.gz

cd tpm-emulator

sudo apt-get install libgmp-dev cmake

./build.sh

cd build

sudo make install

sudo depmod -a

在实验中也需要使用TSS，安装TSS软件栈的指令如下：

sudo apt-get install libtspi-dev trousers

### 3.2 初始化

将源码在Windows下解压然后拷贝到虚拟机中，直接利用Makefile编译本次实验源码。之后正式开始实验，初始化操作的指令和相关说明如下所示：

sudo modprobe tpmd\_dev

sudo tpmd -f -d clear

#再打开一个终端(注意：前面tpmd那个终端不要关)，运行

sudo tcsd

#在执行完sudo tcsd之后终端显示等待指令



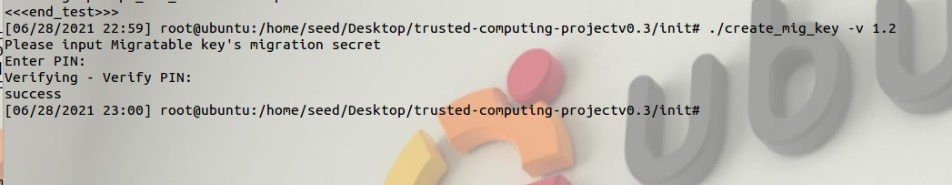
# 确保TPM没有被TakeOwnership，否则会出错

#进入init目录，运行(

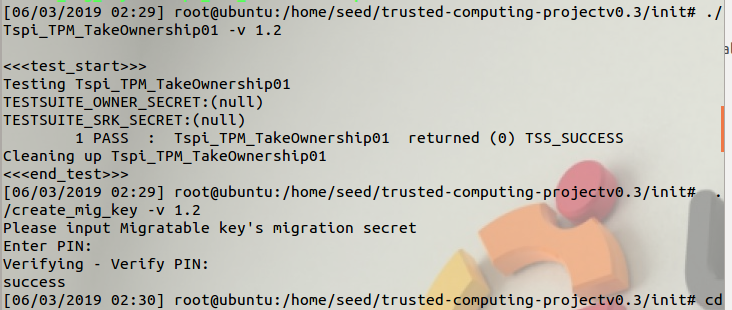
cd /home/seed/trusted-computing-projectv0.3/init)

./Tspi\_TPM\_TakeOwnership01 -v 1.2

#运行 ./create\_mig\_key -v 1.2（输入pin）

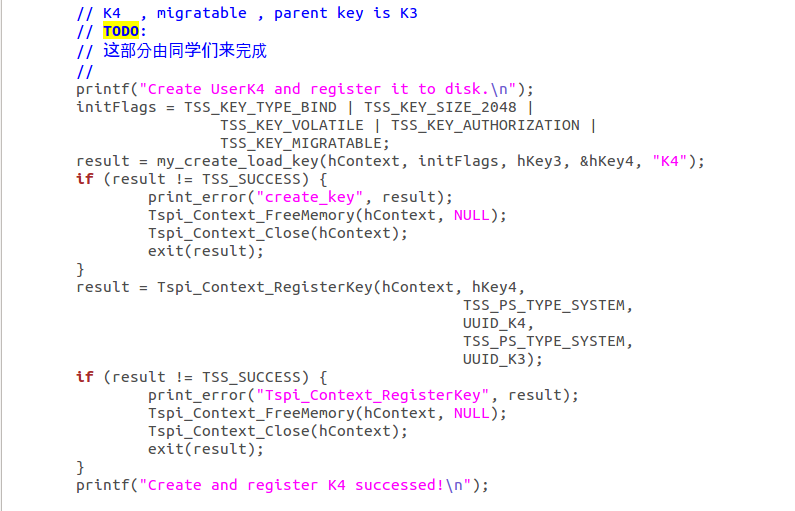


以上所有安装流程完成后，在init阶段的输出结果如下图所示：



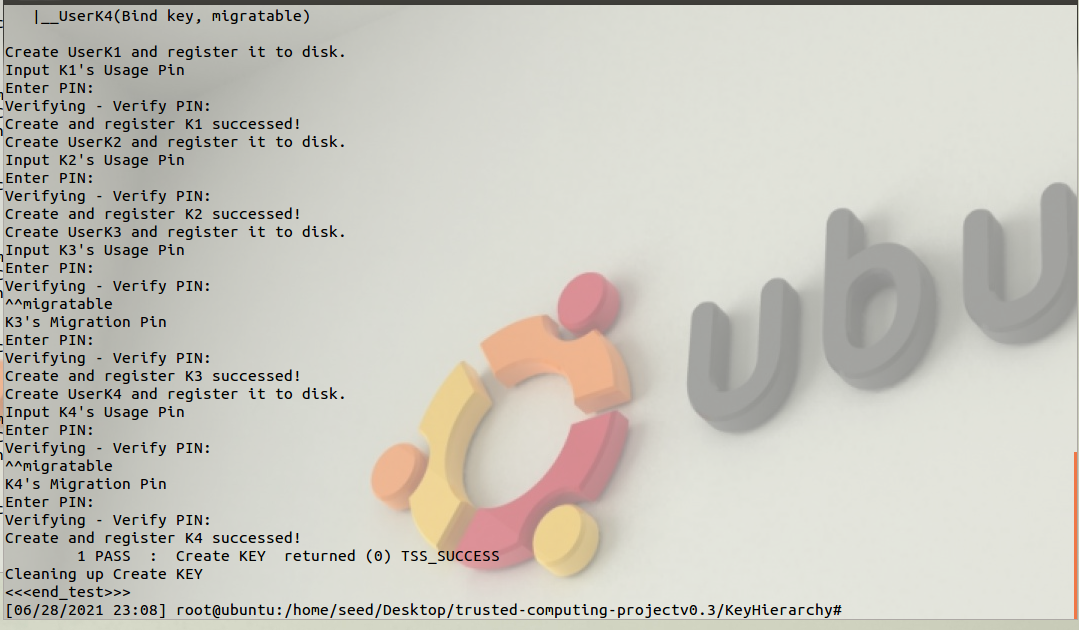
### 3.3 秘钥层次（KeyHierarchy）

进入Key hierarchy目录（seed虚拟机：cd /home/seed/trusted-computing-projectv0.3/KeyHierarchy）参考K1、K2、K3的创建过程，以及TSS文档，完善create\_register\_key.c中创建K4的代码如下：



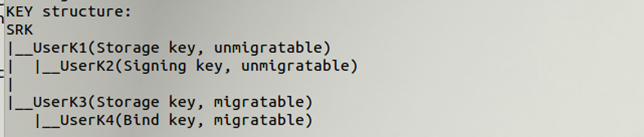
K4的代码与K1、K2、K3创建过程的代码相似，只需要理解函数的意义修改部分的参数即可。

利用Makefile文件生成对应的create\_register\_key可执行程序，而后使用./create\_register\_key -v 1.2指令执行：



设定在实验中需要的各个密钥的值，UserK1是不可迁移的Storage Key，UserK2是不可迁移的Signing Key，UserK3是可迁移的Storage Key，UserK4是可迁移的Bind Key。

该程序的目的就是在加载SRK之后，创建并加载四个register UserKey，将创建的密钥层次结构注册到系统中的永久存储区。



在将密钥存储到系统永久存储区后，要在TPM中进行使用，需要将他们从系统痛就存储区Load到TPM中。load\_key.c的流程就是：

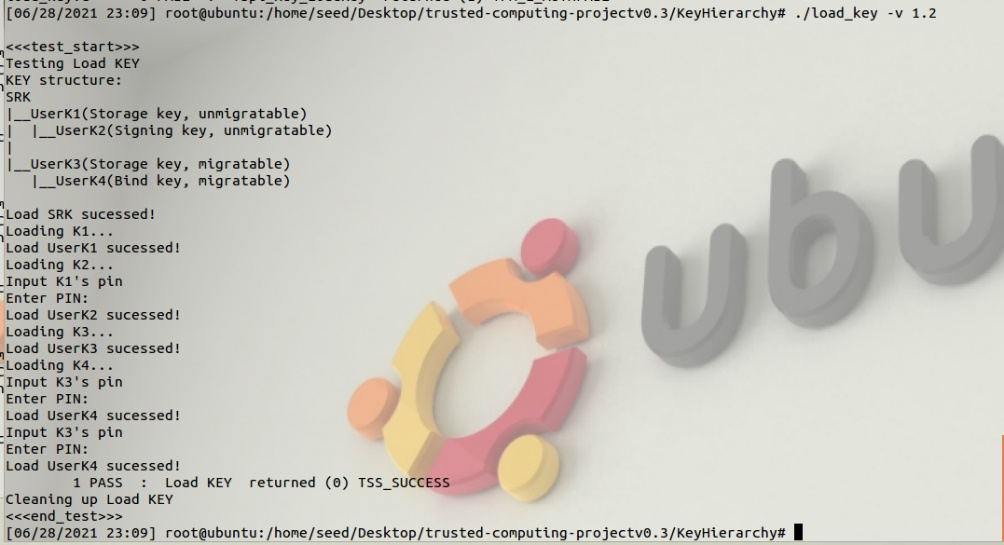
1. load SRK并验证其使用权限
2. load K1的子密钥K2，并验证其使用权限。
3. 重新验证SRK的使用权限(每次使用之前都需验证)。
4. load SRK 的子密钥K3，并验证其使用权限。
5. load K3的子密钥K4，并验证其使用权限。

参考K1、K2、K3的加载过程，以及TSS文档，完善了load\_key.c中加载K4的代码，运行



在上述代码中，首先利用Tspi\_Context\_LoadKeyByUUID()函数，加载密钥，并判断是否加载成功。在使用set\_popup\_secret()函数来验证使用策略，最后给SRK进行授权并验证。验证成功后，即可打印“Load UserK4 sucessed！”。

使用./load\_key -v 1.2执行该程序：



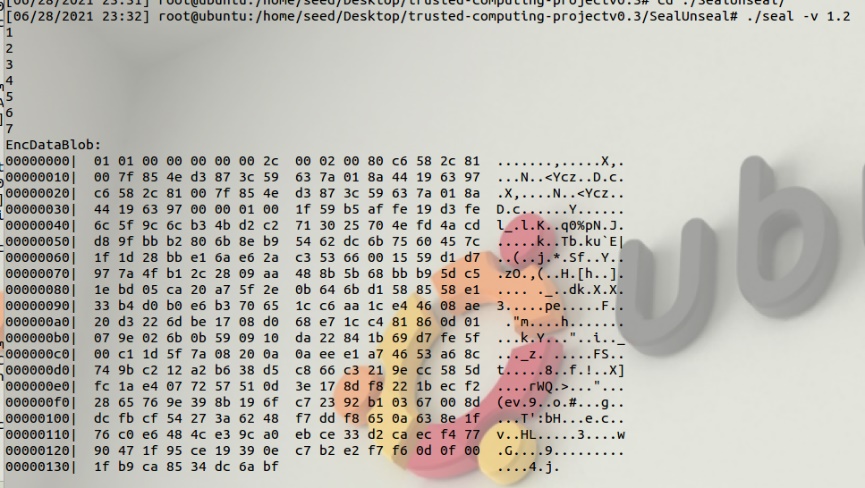
可以看到输入在create\_register\_key.c文件中创建的各个密钥，即可完成这些密钥的加载。

### 3.4 秘钥迁移Seal、Unseal和extend

进入SealUnseal目录(cd /home/seed/trusted-computing-projectv0.3/SealUnseal),实验需要完成unseal\_file.c文件，完成后在trusted-computing-projectv0.3文件夹中make。

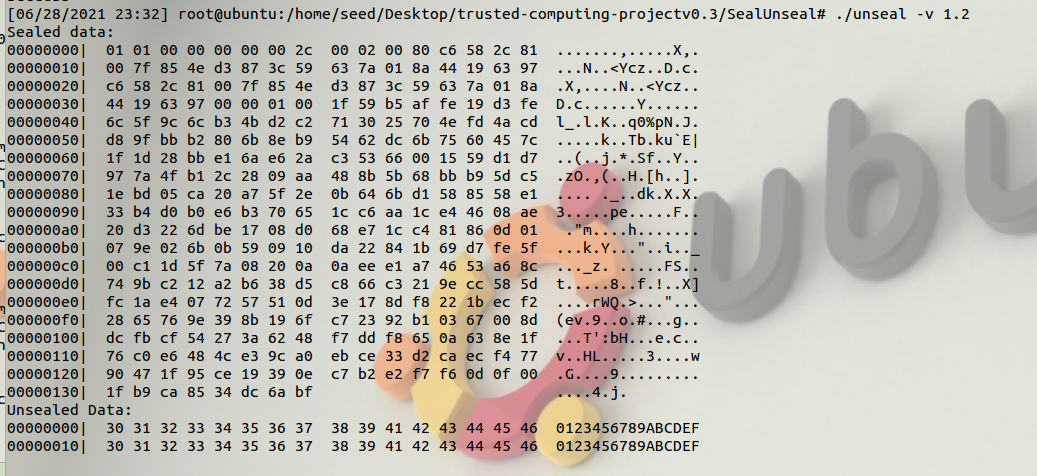
内置了TPM芯片的计算机可以创建一个秘钥，该秘钥被关联到特定的硬件或软件条件，这被称为密封秘钥。首次创建密封秘钥时，TPM将记录配置值和文件哈希的快照。仅当前系统值与快照中的值相匹配时才解密封或释放密封秘钥。简单来说，“密封”是指存储从当前使用的软件、硬件配置衍生出的秘钥，并用这个秘钥加密私有数据，从而实现对它的保护。在编写unseal\_file.c文件后，执行以下指令完成密钥迁移和PCR寄存器的扩展，并查看打印出来的信息：

1. 进入SealUnseal目录(cd /home/seed/trusted-computing-projectv0.3/SealUnseal)
2. 执行 . /seal -v 1.2 指令可以看到执行成功的现象：

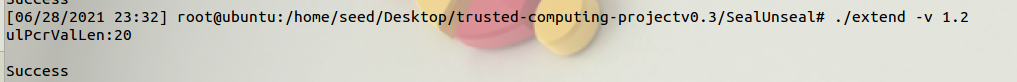


Sealing Key是内置了TPM的计算机可以创建一个密钥，该密钥不仅被绑定，而且还被连接到特定硬件或软件条件，这称为密封密钥。 首次创建密封密钥时，TPM将记录配置值和文件哈希的快照。仅在这些当前系统值与快照中的值相匹配时才解封或释放密封密钥。是不可迁移密钥。

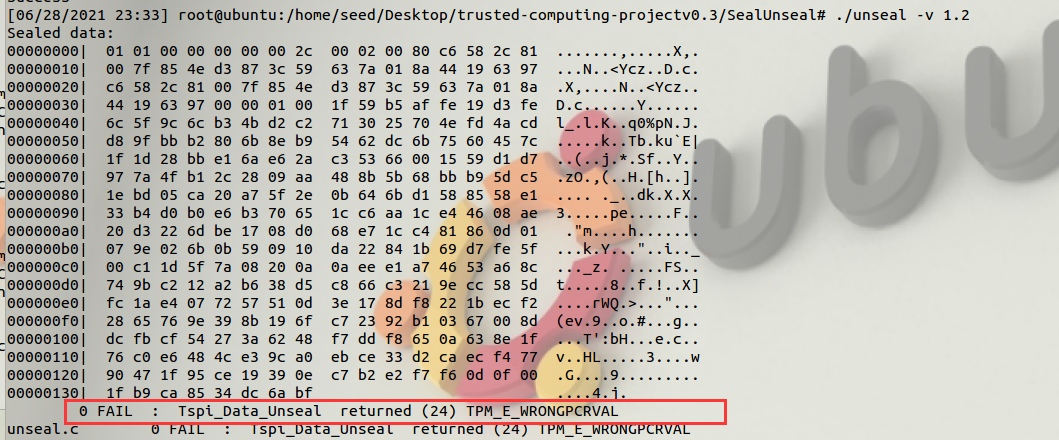
1. 执行 . /unseal -v 1.2 指令，执行成功：



1. 执行 . /extend -v 1.2 指令，实现PCR寄存器的扩展，执行成功：

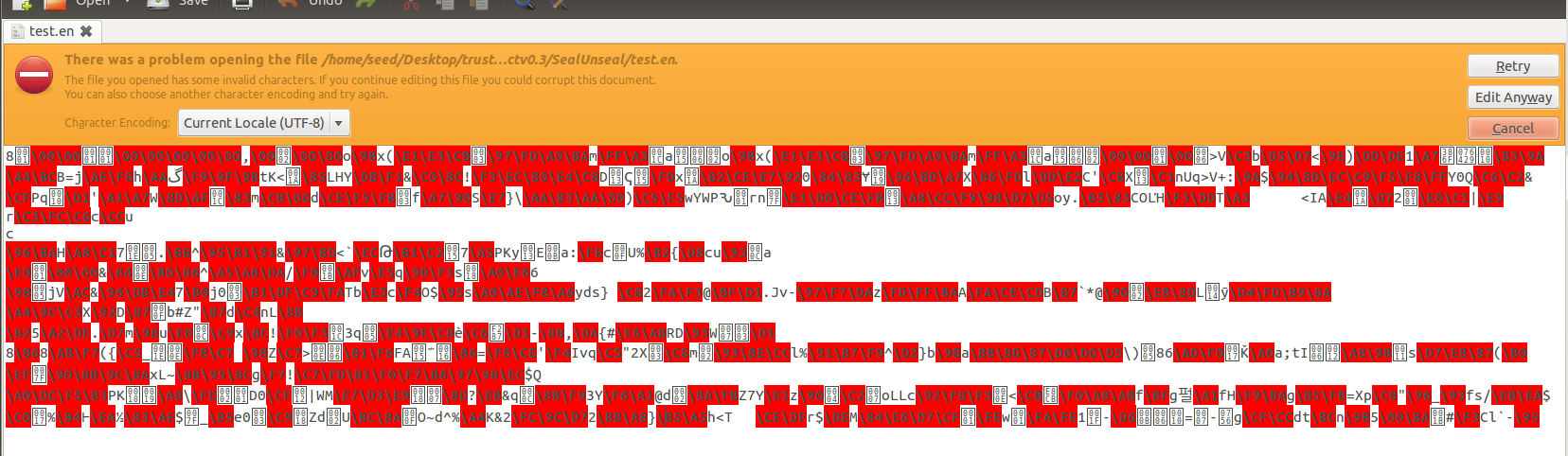


1. 执行 . /unseal -v 1.2 指令，执行失败：



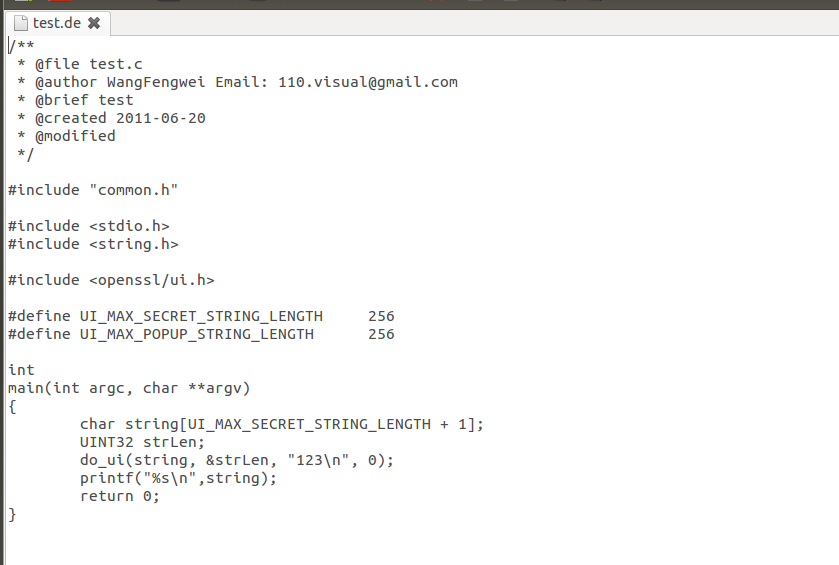
可以看到运行失败的错误码提示，指示TPM\_E\_WRONGPCRVAL错误。

1. 执行 . /seal\_file test.c test.en 指令，查看文件test.en的内容：



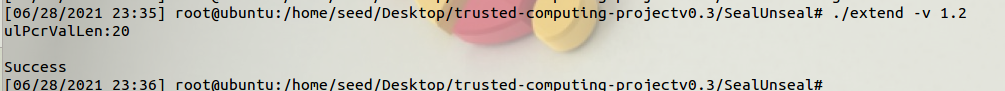
Test.c本身是一个可读的C语言程序，而在使用seal\_file密封之后，test.en的内容变成乱码无法查阅，这是加密后的结果。

1. 执行 . /unseal\_file test.en test.de 指令，查看文件test.de的内容：

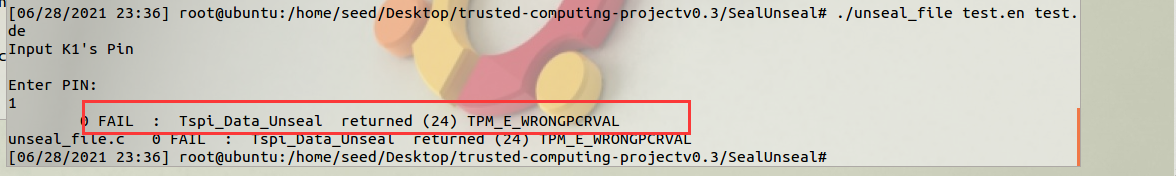


使用seal\_file对test.en进行解密封之后，得到的test.de文件是原始的明文文件。

1. 运行./extend -v 1.2 ，再完成对PCR寄存器的扩展：



1. 运行./unseal\_file test.en test.de，对test.en文件的解密封失败：



### 4.5 秘钥迁移（KeyMigration）

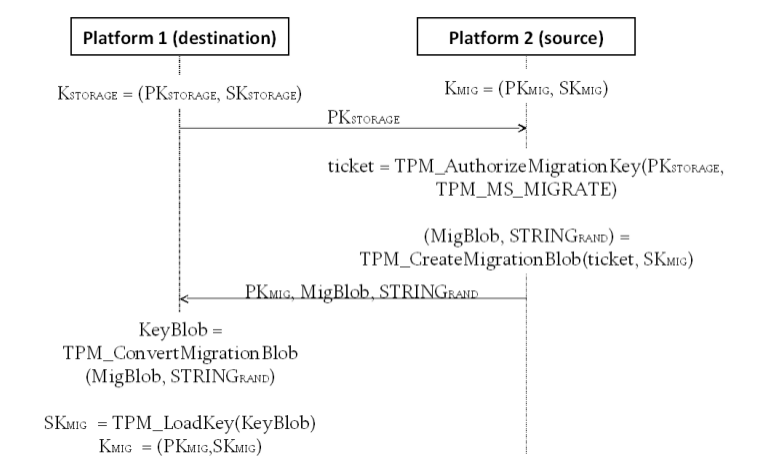
从平台2将秘钥迁移到平台1的过程大致如下：

1. 平台1：将一个存储秘钥对的公钥PKstorage发送到平台2
2. 平台2：将需要移植的秘钥的私钥部分利用PKstorage进行重新加密
3. 平台1：加载平台2发送过来的加密私钥部分

其中，如图相关函数的说明如下：

1. TPM\_AuthorizeMigrationKey:TPM所有者提供要迁移的秘钥的公钥授权。
2. TPM\_CreateMigrationBlob：用KeyBlob封装要迁移的秘钥的私钥部分。
3. TPM\_LoadKey:TPM从KeyBlob中解密私钥部分。

密钥的迁移过程图示如下，该节任务就是补全迁移到Blob的代码：



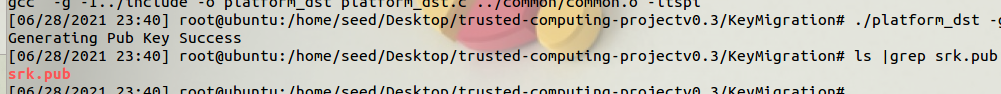
进入Key Migration目录(cd /home/seed/trusted-computing-projectv0.3/KeyMigration)，完成platform\_dst.c中的部分代码。

在platform\_dst.c文件中需要添加的代码部分就是将密钥迁移给Bob，在参考了Tspi\_Key\_ ConvertMigrationBlob的代码后，完成编写了下图的代码：

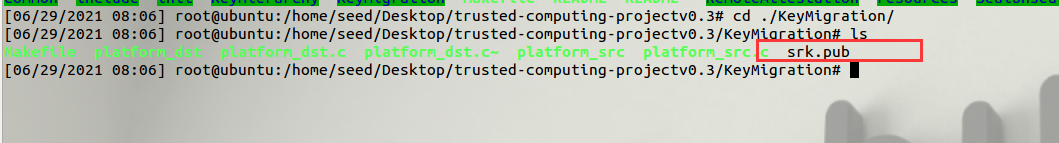


如上图的代码所示，该过程首先利用Tspi\_Key\_ConvertMigrationBlob()函数进行迁移，而后加载新的迁移密钥并验证是否加载成功，再验证新的迁移密钥是否有效。

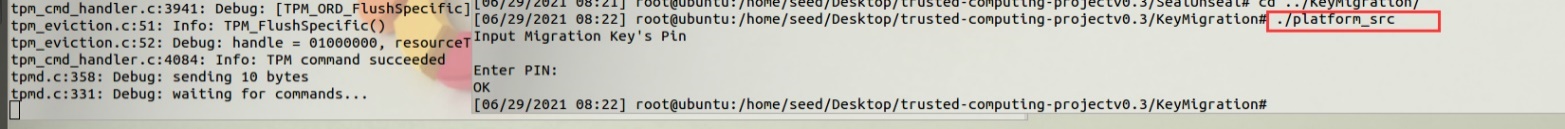
在机器1中运行 ./platform\_dst -g 指令，回产生一个srk.pub的文件：



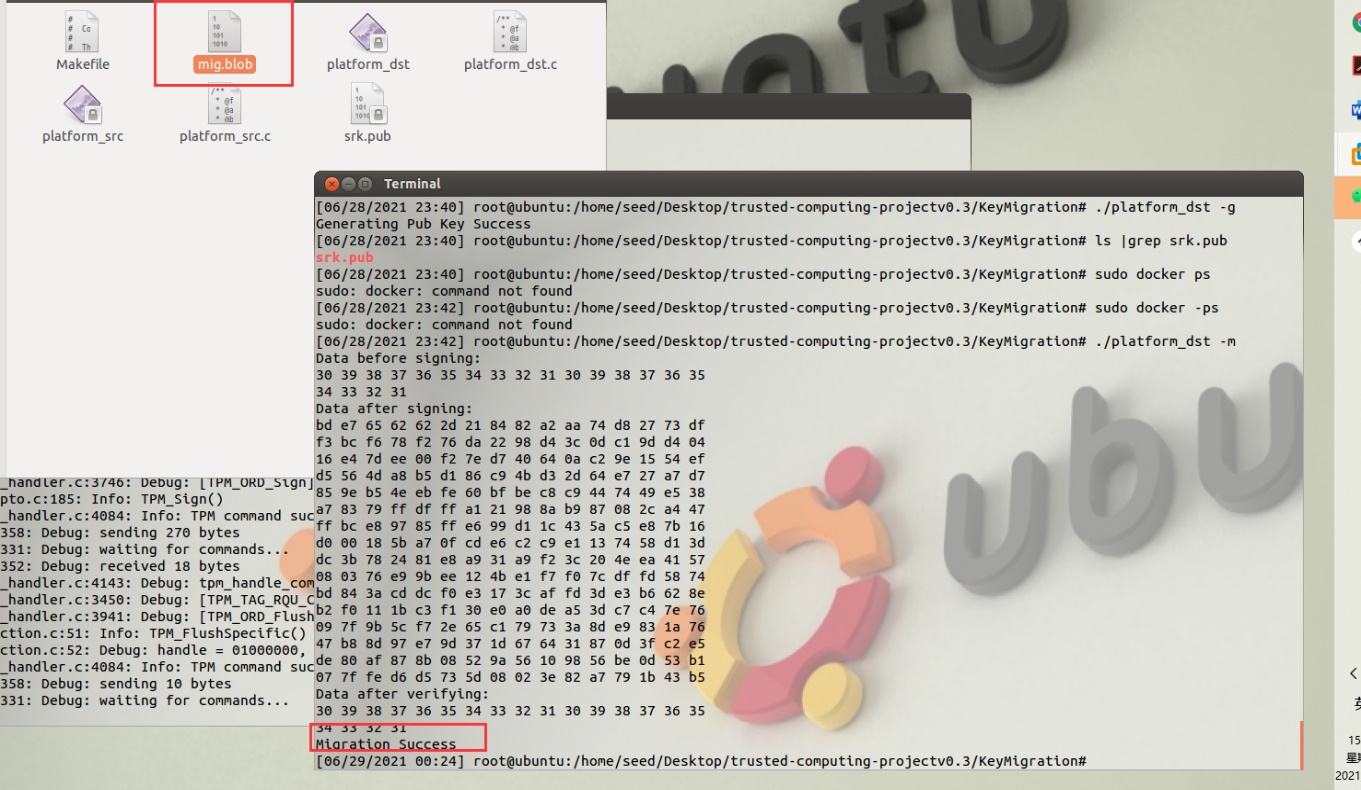
将文件srk.pub拷贝到机器2中：



运行./platform\_src 指令：



执行完指令后会产生名为mig.blob的文件，文件mig.blob拷回到机器1中，在机器1中运行：



在终端打印出了Migration Success的字符串，表示密钥迁移成功！

### 3.6 远程证明（RemoteAttestation）

远程证明的基本流程如下图所示：

1. 首先由挑战者向被证实平台提出证实请求，包括一个随机数用来防止重放攻击；

2. 被证实平台随之搜集在TPM制造时写入的背书证书EK（用来标识可信平台的唯一身份）等等，并用EK生成平台身份密钥AIK来避免暴露隐私，然后将EK、AIK等等发送给隐私签证机构（PCA）；

3. PCA通过验证EK，验证TPM芯片的合法性，并为之颁发身份认证证书，承认AIK的合法性；

4. 被证实平台通过TPM\_Quote操作，使用平台身份密钥（AIK）对PCR值进行签名，并随之将签名值与度量日志和AIK证书发送给挑战者；

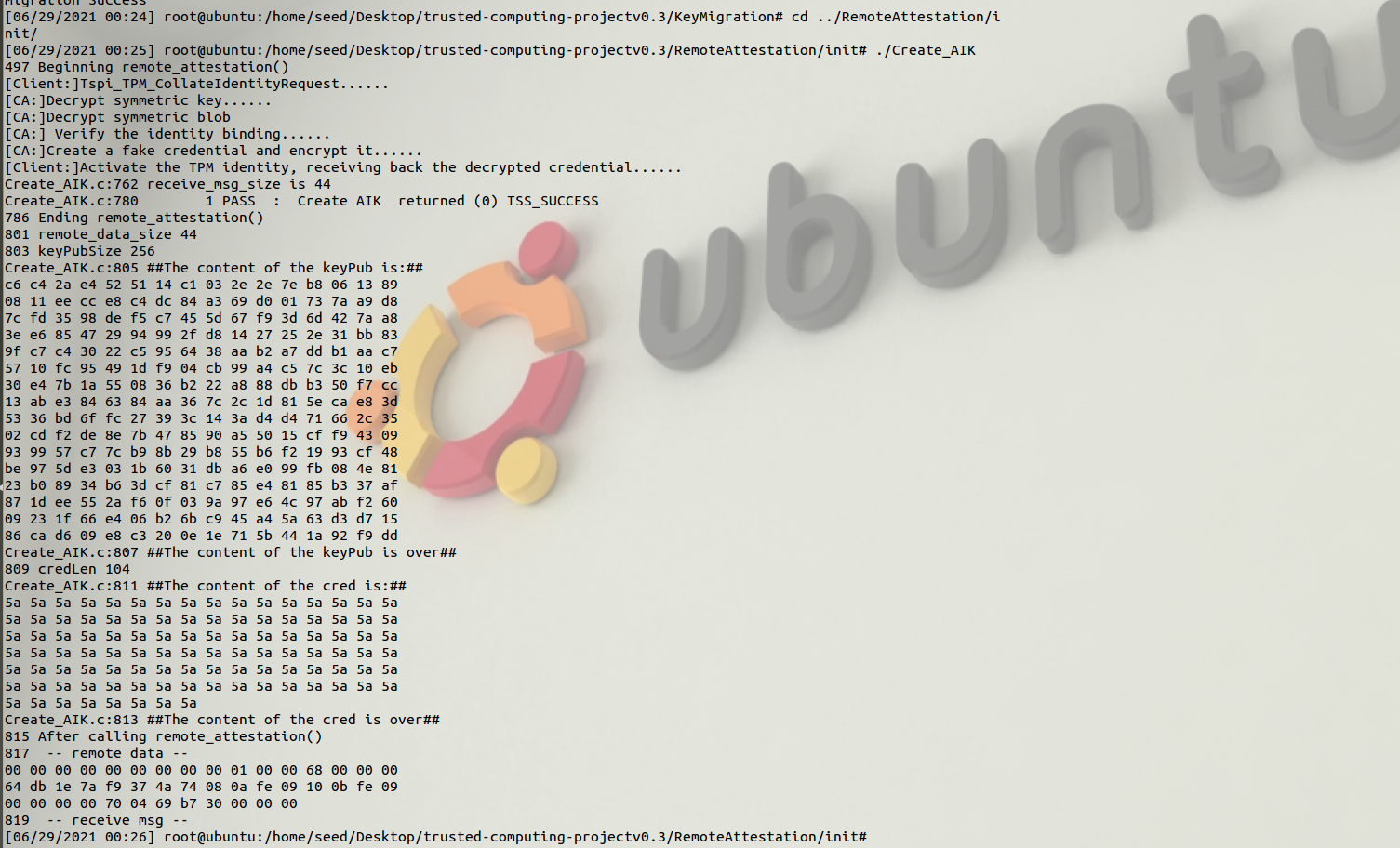
5. 挑战者首先验证AIK证书的有效性，使用AIK公钥对数据进行解密获取PCR，通过PCR确保度量日志真实可信地返回给了挑战者，随后则将度量日志的每一项与预期值进行比对，判断平台是否可信。

以下为实验实现过程：

在机器1中执行以下步骤：

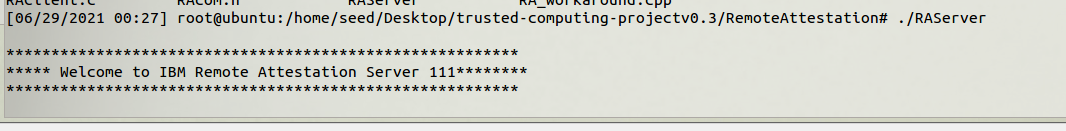
1、进入Remote Attestation\init目录

2、运行./Create\_AIK



3、返回上级目录

4、运行./RAServer

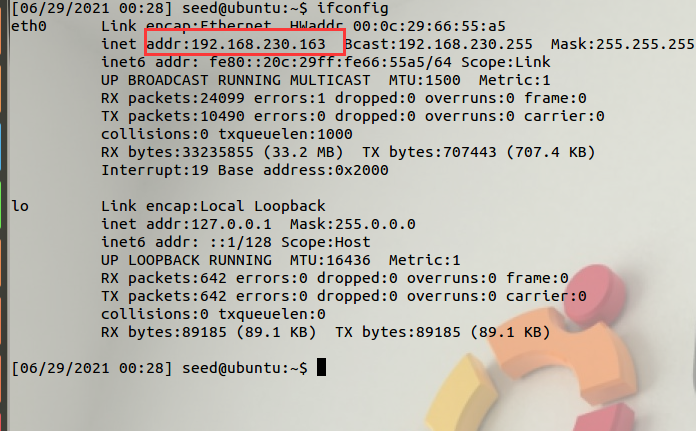


在机器2中执行以下步骤：

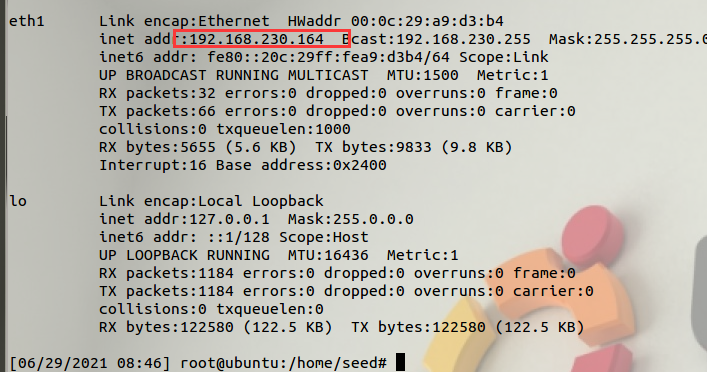
1、进入Remote Attestation目录

2、运行./RAClient 机器2的ip 机器1的ip（如，./RAClient 192.168.200.1 192.168.200.2）

机器1的ip：192.168.230.163



机器2的ip：192.168.230.164



执行指令./RAClient 192.168.230.163 192.168.230.164



实现了远程证明环节。

# 4 问题与解决方案

**问题1：**

下载的Seed Ubantu是助教发送在群中的版本，Vmware Tools的版本不够，将tpm-emulator.tar.gz移入虚拟机需需要更新VmwareTools。

**解决方案：**

参考链接:<https://blog.csdn.net/weixin_33969116/article/details/91555608>

完成VmwareTools的更新。

**问题2：**

在编写密钥迁移的代码时并没有很好的理解密钥迁移的过程，在填写一个参数的时候填写错误。

**解决方案：**

参考查看Tspi\_Key\_ ConvertMigrationBlob的代码，在网上查阅相应的资料发现了错误，需要填写的参数应该是hSRK。

# 5 实验心得与体会

本次实验的在初期的编写中并没有遇到什么问题。在create\_register\_key.c文件中编写创建UserK4的代码，有着K1、K2、K3的代码参考，而且这些密钥的创建过程都很相似，所以完成起来还是比较容易的。Load\_key.c文件的编写也是类似的，参照着其他几个密钥的加载过程编写K4的相应代码。

在完成unseal\_file.c的文件的时候才遇到了一些困难，一开始不知道如何动手。而后想到密封和解封是一组的，去查看了一下seal\_file.c的内容，理解其的加密过程，再反向的进行解密就是unseal.c文件的主要内容。利用test.c文件进行测试，在seal\_file.c文件的密封后，使用unseal去解密封test.en就能得到原始的test.c文件的内容。

在最后的密钥迁移中和远程证明中，再次回顾理解了一下老师上课的内容，对于密钥迁移的过程更加熟悉了。从平台2将秘钥迁移到平台1的过程大致如下：平台1将一个存储秘钥对的公钥PKstorage发送到平台2，平台2再将需要移植的秘钥的私钥部分利用PKstorage进行重新加密。平台1加载平台2发送过来的加密私钥部分，用KeyBlob封装要迁移的秘钥的私钥部分，最后TPM从KeyBlob中解密私钥部分。

本次的实验难度正常，而且将代老师上课讲述的知识实践了一遍，个人感觉收获颇丰，对于TPM各个功能有了更深刻的理解。