1. 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。

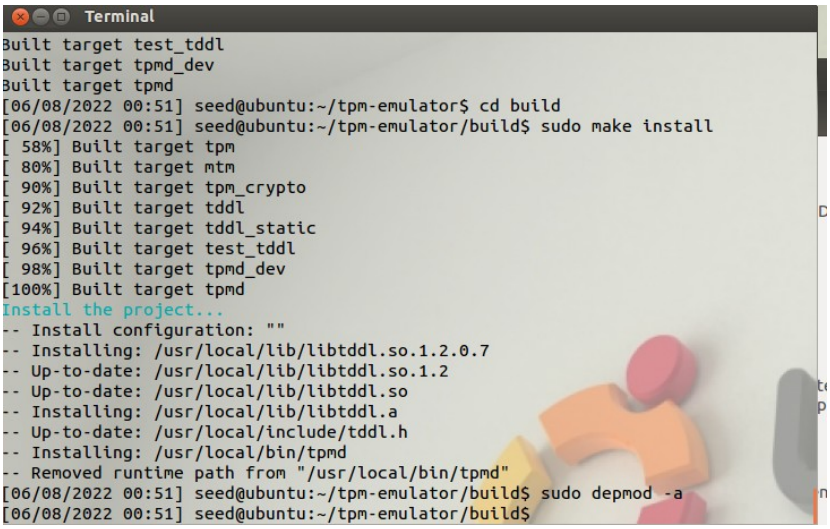
2. 实验环境

- Seed Ubuntu 12.04 LTS 32 位的 VMware 虚拟机
- TPM Emulator
- Trousers

3. 实验过程

3.1 编译

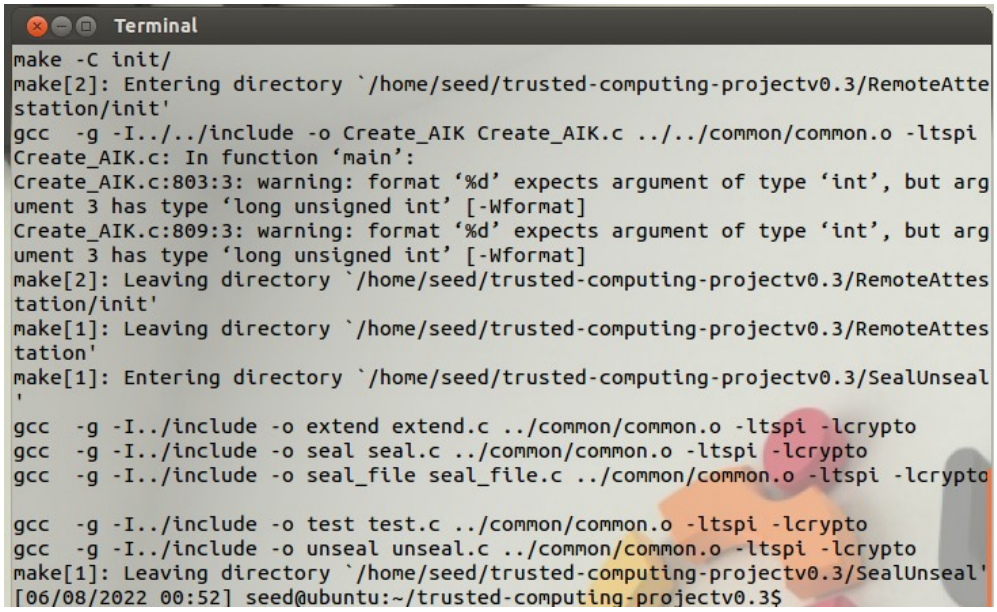
解压并编译安装 TPM Emulator



```
Terminal
Built target test_tddl
Built target tpmd_dev
Built target tpmd
[06/08/2022 00:51] seed@ubuntu:~/tpm-emulator$ cd build
[06/08/2022 00:51] seed@ubuntu:~/tpm-emulator/build$ sudo make install
[ 58%] Built target tpm
[ 80%] Built target mtm
[ 90%] Built target tpm_crypto
[ 92%] Built target tddl
[ 94%] Built target tddl_static
[ 96%] Built target test_tddl
[ 98%] Built target tpmd_dev
[100%] Built target tpmd
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/libtddl.so.1.2.0.7
-- Up-to-date: /usr/local/lib/libtddl.so.1.2
-- Up-to-date: /usr/local/lib/libtddl.so
-- Installing: /usr/local/lib/libtddl.a
-- Up-to-date: /usr/local/include/tddl.h
-- Installing: /usr/local/bin/tpmd
-- Removed runtime path from "/usr/local/bin/tpmd"
[06/08/2022 00:51] seed@ubuntu:~/tpm-emulator/build$ sudo depmod -a
[06/08/2022 00:51] seed@ubuntu:~/tpm-emulator/build$
```

3.2 初始化

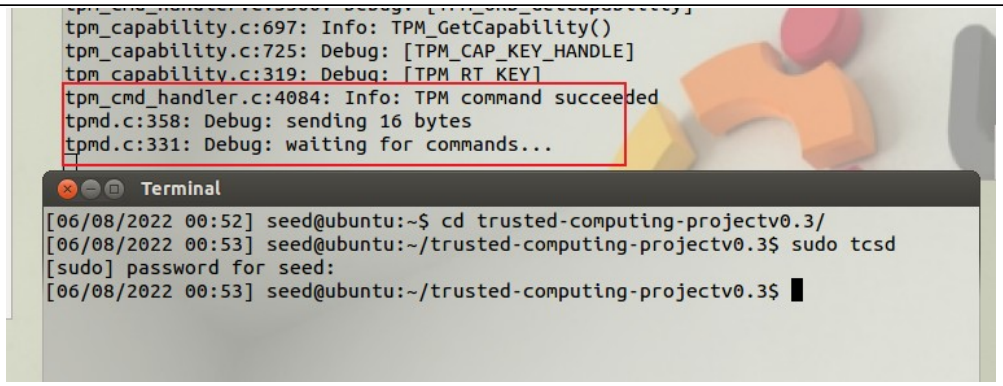
将源码在 Windows 下解压然后拷贝到虚拟机中，进入目录，执行 sudo make



```
Terminal
make -C init/
make[2]: Entering directory `/home/seed/trusted-computing-projectv0.3/RemoteAtte
station/init'
gcc  -g -I../../include -o Create_AIK Create_AIK.c ../../common/common.o -ltspi
Create_AIK.c: In function ‘main’:
Create_AIK.c:803:3: warning: format ‘%d’ expects argument of type ‘int’, but arg
ument 3 has type ‘long unsigned int’ [-Wformat]
Create_AIK.c:809:3: warning: format ‘%d’ expects argument of type ‘int’, but arg
ument 3 has type ‘long unsigned int’ [-Wformat]
make[2]: Leaving directory `/home/seed/trusted-computing-projectv0.3/RemoteAttes
tation/init'
make[1]: Leaving directory `/home/seed/trusted-computing-projectv0.3/RemoteAttes
tation'
make[1]: Entering directory `/home/seed/trusted-computing-projectv0.3/SealUnseal
'
gcc  -g -I../include -o extend extend.c ../common/common.o -ltspi -lcrypto
gcc  -g -I../include -o seal seal.c ../common/common.o -ltspi -lcrypto
gcc  -g -I../include -o seal_file seal_file.c ../common/common.o -ltspi -lcrypto

gcc  -g -I../include -o test test.c ../common/common.o -ltspi -lcrypto
gcc  -g -I../include -o unseal unseal.c ../common/common.o -ltspi -lcrypto
make[1]: Leaving directory `/home/seed/trusted-computing-projectv0.3/SealUnseal'
[06/08/2022 00:52] seed@ubuntu:~/trusted-computing-projectv0.3$
```

之后正式开始实验，初始化操作的指令如下

```
sudo modprobe tpmd_dev
sudo tpmd -f -d clear

#再打开一个终端(注意：前面 tpmd 那个终端一直不要关)，运行

sudo tcsd

#在执行完 sudo tcsd 之后终端显示等待指令
```

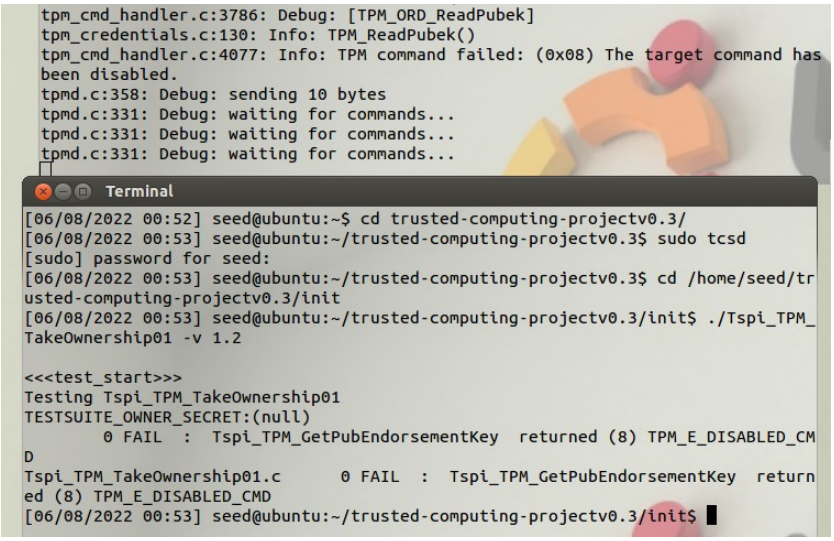


确保 TPM 没有被 TakeOwnership，否则会出错

#进入 init 目录

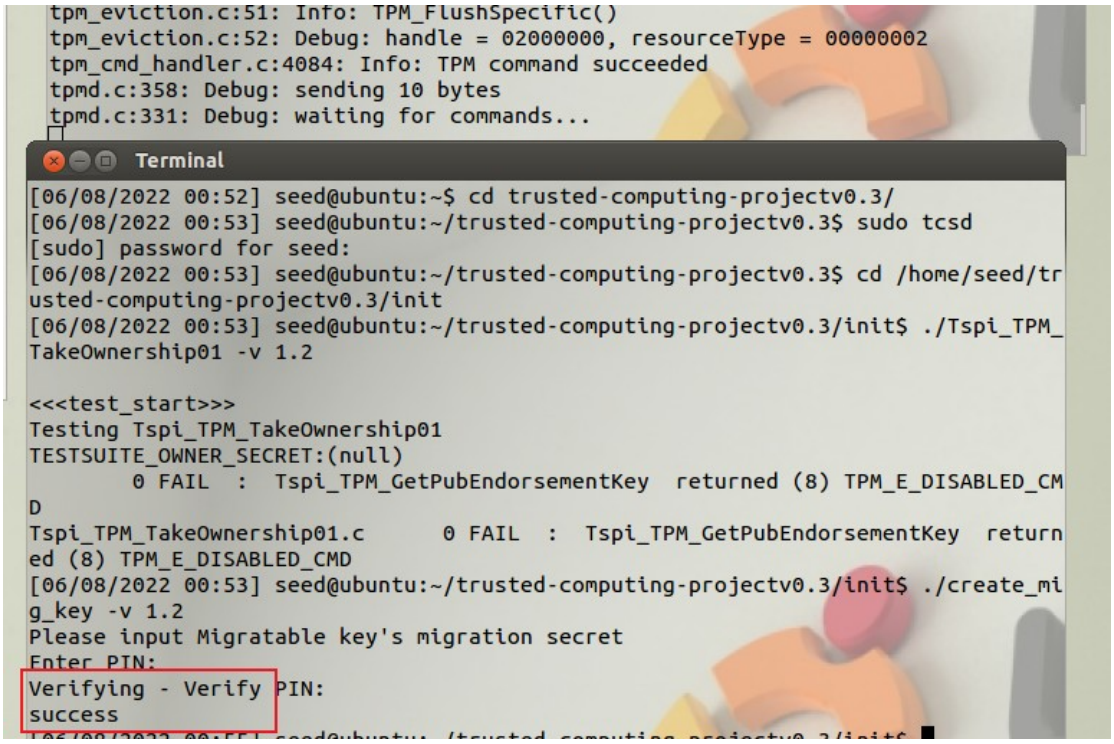
(运行 cd /home/seed/trusted-computing-projectv0.3/init)

执行 ./Tspi_TPM_TakeOwnership01 -v 1.2



执行 ./create_mig_key -v 1.2（输入 PIN） PIN 为 123456

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）

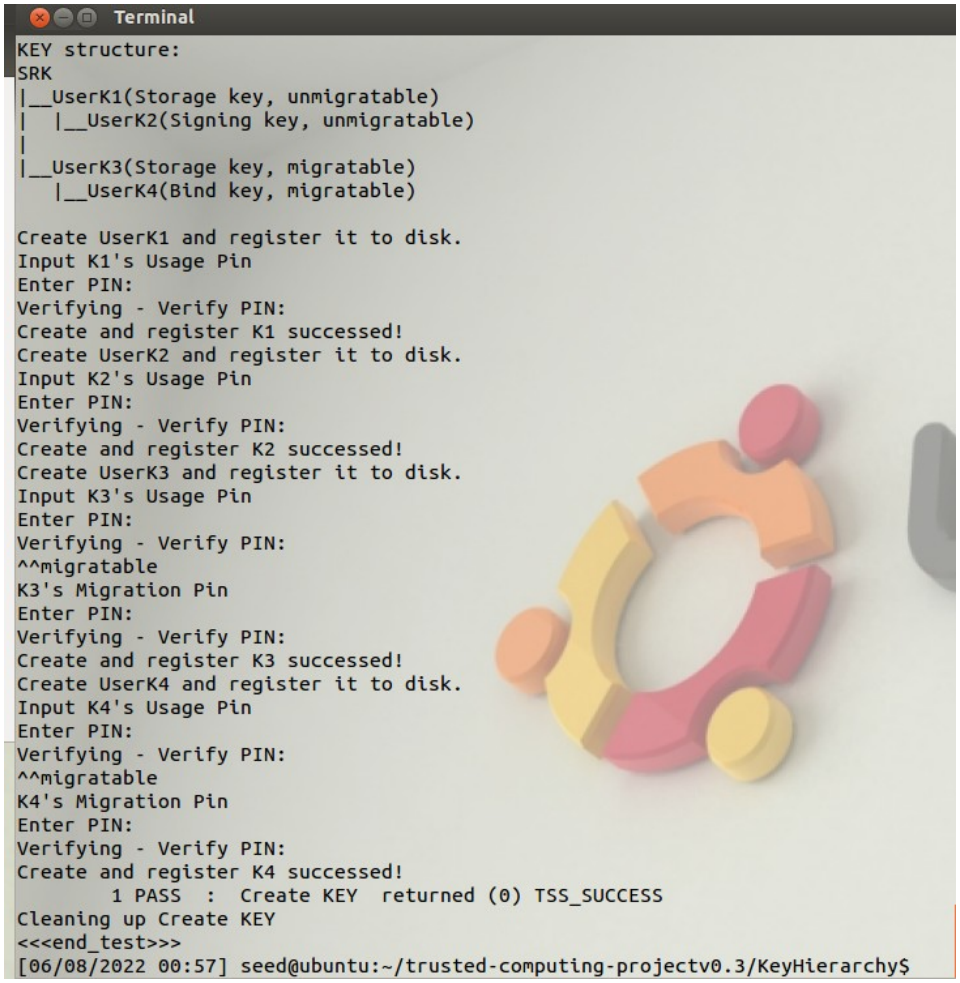
进入 Key hierarchy 目录（seed 虚拟机： cd /home/seed/Desktop/lab/trusted-computing-projectv0.3/KeyHierarchy）

完善 create_register_key.c 中创建 K4 的代码如下：

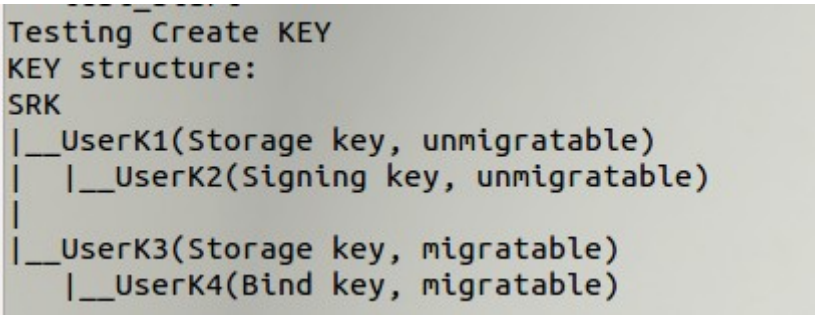
```
        exit(result);
}
printf("Create and register K3 successful!\n");

// K4 , migratable , parent key is K3
// TODO:
// 这部分由同学们来完成
//
printf("Create UserK4 and register it to disk.\n");

initFlags = TSS_KEY_TYPE_BIND | TSS_KEY_SIZE_2048 |
                                TSS_KEY_VOLATILE | TSS_KEY_AUTHORIZATION|
                                TSS_KEY_MIGRATABLE;
result = my_create_load_key(hContext, initFlags,hKey3,&hKey4,"K4");
if (result != TSS_SUCCESS) {
        print_error("create_key",result);
        Tspi_Context_FreeMemory(hContext,NULL);
        Tspi_Context_Close(hContext);
        exit(result);
}

result = Tspi_Context_RegisterKey(hContext,hKey4,TSS_PS_TYPE_SYSTEM,UUID_K4,TSS_PS_TYPE_SYSTEM,UUID_K3);
if (result != TSS_SUCCESS) {
        print_error("Tspi_Context_RegisterKey",result);
        Tspi_Context_FreeMemory(hContext,NULL);
        Tspi_Context_Close(hContext);
        exit(result);
}


printf("Create and register K4 successful!\n");
```

执行 sudo make 生成对应的 create_register_key

使用./create_register_key -v 1.2 指令执行：



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。



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，sudo make 编译

```
// ------------------------------- load k4 --------------------------------
// TODO:
//      这部分代码由同学们自己完成
//
printf("Loading K4...\n");

result = Tspi_Context_GetKeyByUUID(hContext, TSS_PS_TYPE_SYSTEM, UUID_K4, &hKey4);
if (result != TSS_SUCCESS) {
    print_error("Tspi_Context_GetKeyByUUID", result);
    print_error_exit(nameOfFunction, err_string(result));
    Tspi_Context_FreeMemory(hContext, NULL);
    Tspi_Context_Close(hContext);
    exit(result);
}

result = set_popup_secret(hContext, hKey3, TSS_POLICY_USAGE, "Input K3's pin", 0);
if (TSS_SUCCESS != result) {
    print_error("set_popup_secret", result);
    Tspi_Context_Close(hContext);
    return result;
}

result = Tspi_Key_LoadKey(hKey4, hKey3);
if (result != TSS_SUCCESS) {
    print_error("Tspi_Key_LoadKey", result);
    Tspi_Context_FreeMemory(hContext, NULL);
    Tspi_Context_Close(hContext);
    exit(result);
}

printf("Load UserK4 sucessed!\n");
```

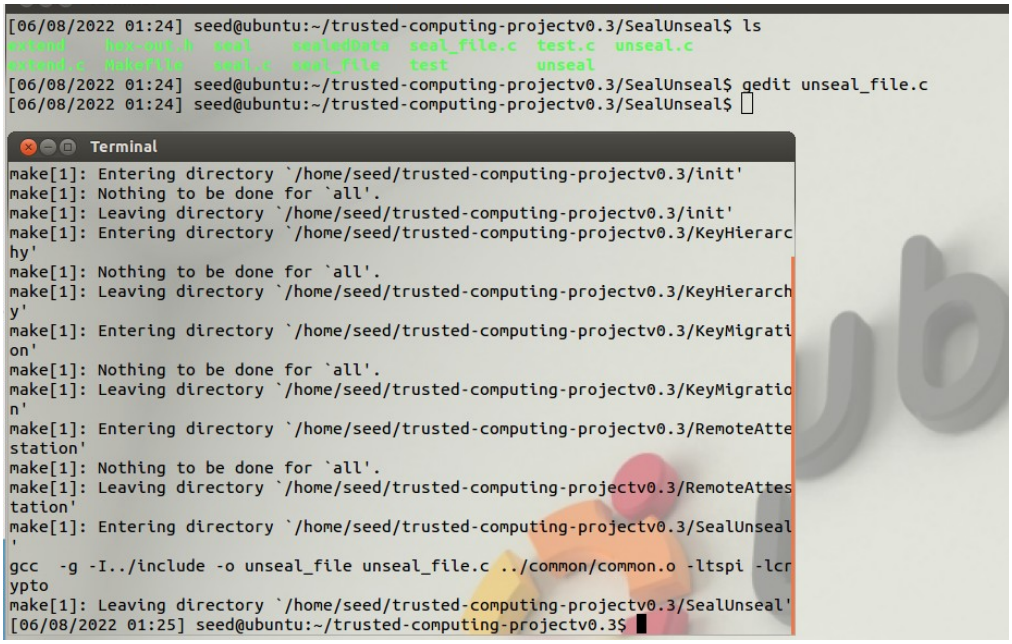
使用./load_key -v 1.2 执行该程序：

```
[06/08/2022 01:19] seed@ubuntu:~/trusted-computing-projectv0.3/KeyHierarchy$ ./l
oad_key -v 1.2

<<<test_start>>>
Testing Load KEY
KEY structure:
SRK
|__UserK1(Storage key, unmigratable)
|   |__UserK2(Signing key, unmigratable)
|
|__UserK3(Storage key, migratable)
    |__UserK4(Bind key, migratable)

Load SRK sucessed!
Loading K1...
Load UserK1 sucessed!
Loading K2...
Input K1's pin
Enter PIN:
Load UserK2 sucessed!
Loading K3...
Load UserK3 sucessed!
Loading K4...
Input K3's pin
Enter PIN:
Load UserK4 sucessed!
Input K3's pin
Enter PIN:
Load UserK4 sucessed!
        1 PASS  :  Load KEY  returned (0) TSS_SUCCESS
Cleaning up Load KEY
<<<end_test>>>
[06/08/2022 01:19] seed@ubuntu:~/trusted-computing-projectv0.3/KeyHierarchy$
```

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 sudo make。



然后:

进入 SealUnseal 目录

(cd /home/seed/trusted-computing-projectv0.3/SealUnseal)

执行 ./seal -v 1.2 指令可以看到执行成功的现象:

```
[06/08/2022 01:24] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./seal -v 1.2
1
2
3
4
5
6
7
EncDataBlob:
00000000|  01 01 00 00 00 00 00 2c  00 02 00 80 c6 58 2c 81  .......,.....X,.
00000010|  00 7f 85 4e d3 87 3c 59  63 7a 01 8a 44 19 63 97  ...N..<Ycz..D.c.
00000020|  c6 58 2c 81 00 7f 85 4e  d3 87 3c 59 63 7a 01 8a  .X,....N..<Ycz..
00000030|  44 19 63 97 00 00 01 00  64 aa 0d a5 7d a7 47 8a  D.c.....d...}.G.
00000040|  d3 de 8a bf ce 26 21 92  3c 5f 99 3e c8 4d 7d 00  .....&!.<_.>.M}.
00000050|  b7 09 9d 4f 78 65 95 a4  96 83 66 0c 79 47 4a 5e  ...Oxe....f.yGJ^
00000060|  e0 12 04 c0 c8 4a ac e6  43 91 c0 28 8f 4a 15 2c  .....J..C..(.J.,
00000070|  f9 7c 6e 9c ae 85 28 03  0b 4e e7 2d de a1 a5 ae  .|n...(..N.-....
00000080|  83 53 2b 95 26 c2 9e 32  68 44 f9 64 a5 00 f0 3d  .S+.&..2hD.d...=
00000090|  f3 f7 1d e4 30 fd 03 d2  2b 7c ca 5f 11 48 26 16  ....0...+|._.H&.
000000a0|  09 53 b4 99 89 06 b6 54  3c b9 f2 66 34 a8 35 20  .S.....T<..f4.5
000000b0|  b5 97 8c 1f ec 8e b9 ce  d2 9d ee 72 c4 68 0a 8c  ...........r.h..
000000c0|  8a 54 64 ed 2f c2 39 ab  15 4b f2 cb 91 29 05 fb  .Td./.9..K...)..
000000d0|  26 8b 9e 9d 4d 55 9b c8  b7 6a 4e 49 60 2c 4a 1f  &...MU...jNI`,J.
000000e0|  ef ef ed c1 f3 ab 0e 0b  c0 02 e2 6e 38 e2 6b 3b  ...........n8.k;
000000f0|  a6 ed 93 c2 89 e5 48 46  0c 3e d1 b1 c2 57 c8 e7  ......HF.>...W..
00000100|  a6 b4 c0 22 f5 a4 b4 f2  8f 9a b1 54 7a a7 0a f9  ..."......Tz...
00000110|  03 bb ce 28 f0 12 83 59  0c 7f c0 14 e7 4c bd 77  ...(...Y.....L.w
00000120|  be c9 de a2 30 9e 14 fb  d2 24 76 61 a7 53 e4 f6  ....0....$va.S..
00000130|  2b b7 00 24 2f db aa b8                           +..$/...

Success
```

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。

之后执行：

1. 执行 ./unseal -v 1.2 指令，执行成功：

```
[06/08/2022 01:25] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./unseal -v 1.2
Sealed data:
00000000|  01 01 00 00 00 00 00 2c  00 02 00 80 c6 58 2c 81  .......,.....X,.
00000010|  00 7f 85 4e d3 87 3c 59  63 7a 01 8a 44 19 63 97  ...N..<Ycz..D.c.
00000020|  c6 58 2c 81 00 7f 85 4e  d3 87 3c 59 63 7a 01 8a  .X,....N..<Ycz..
00000030|  44 19 63 97 00 00 01 00  64 aa 0d a5 7d a7 47 8a  D.c.....d...}.G.
00000040|  d3 de 8a bf ce 26 21 92  3c 5f 99 3e c8 4d 7d 00  .....&!.<_.>.M}.
00000050|  b7 09 9d 4f 78 65 95 a4  96 83 66 0c 79 47 4a 5e  ...Oxe....f.yGJ^
00000060|  e0 12 04 c0 c8 4a ac e6  43 91 c0 28 8f 4a 15 2c  .....J..C..(.J.,
00000070|  f9 7c 6e 9c ae 85 28 03  0b 4e e7 2d de a1 a5 ae  .|n...(..N.-....
00000080|  83 53 2b 95 26 c2 9e 32  68 44 f9 64 a5 00 f0 3d  .S+.&..2hD.d...=
00000090|  f3 f7 1d e4 30 fd 03 d2  2b 7c ca 5f 11 48 26 16  ....0...+|._.H&.
000000a0|  09 53 b4 99 89 06 b6 54  3c b9 f2 66 34 a8 35 20  .S.....T<..f4.5
000000b0|  b5 97 8c 1f ec 8e b9 ce  d2 9d ee 72 c4 68 0a 8c  ...........r.h..
000000c0|  8a 54 64 ed 2f c2 39 ab  15 4b f2 cb 91 29 05 fb  .Td./.9..K...)..
000000d0|  26 8b 9e 9d 4d 55 9b c8  b7 6a 4e 49 60 2c 4a 1f  &...MU...jNI`,J.
000000e0|  ef ef ed c1 f3 ab 0e 0b  c0 02 e2 6e 38 e2 6b 3b  ...........n8.k;
000000f0|  a6 ed 93 c2 89 e5 48 46  0c 3e d1 b1 c2 57 c8 e7  ......HF.>...W..
00000100|  a6 b4 c0 22 f5 a4 b4 f2  8f 9a b1 54 7a a7 0a f9  ..."......Tz...
00000110|  03 bb ce 28 f0 12 83 59  0c 7f c0 14 e7 4c bd 77  ...(...Y.....L.w
00000120|  be c9 de a2 30 9e 14 fb  d2 24 76 61 a7 53 e4 f6  ....0....$va.S..
00000130|  2b b7 00 24 2f db aa b8                           +..$/...
Unsealed Data:
00000000|  30 31 32 33 34 35 36 37  38 39 41 42 43 44 45 46  0123456789ABCDEF
00000010|  30 31 32 33 34 35 36 37  38 39 41 42 43 44 45 46  0123456789ABCDEF

Success
[06/08/2022 01:25] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$
```

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

```
[06/08/2022 01:25] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./extend -v 1.2
ulPcrValLen:20

Success
```

3. 执行 ./unseal -v 1.2 指令，执行失败：

```
[06/08/2022 01:26] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./unseal -v 1.2
Sealed data:
00000000|  01 01 00 00 00 00 00 2c  00 02 00 80 c6 58 2c 81  .......,.....X,.
00000010|  00 7f 85 4e d3 87 3c 59  63 7a 01 8a 44 19 63 97  ...N..<Ycz..D.c.
00000020|  c6 58 2c 81 00 7f 85 4e  d3 87 3c 59 63 7a 01 8a  .X,....N..<Ycz..
00000030|  44 19 63 97 00 00 01 00  64 aa 0d a5 7d a7 47 8a  D.c.....d...}.G.
00000040|  d3 de 8a bf ce 26 21 92  3c 5f 99 3e c8 4d 7d 00  .....&!.<_.>.M}.
00000050|  b7 09 9d 4f 78 65 95 a4  96 83 66 0c 79 47 4a 5e  ...Oxe....f.yGJ^
00000060|  e0 12 04 c0 c8 4a ac e6  43 91 c0 28 8f 4a 15 2c  .....J..C..(.J.,
00000070|  f9 7c 6e 9c ae 85 28 03  0b 4e e7 2d de a1 a5 ae  .|n...(..N.-....
00000080|  83 53 2b 95 26 c2 9e 32  68 44 f9 64 a5 00 f0 3d  .S+.&..2hD.d...=
00000090|  f3 f7 1d e4 30 fd 03 d2  2b 7c ca 5f 11 48 26 16  ....0...+|._.H&.
000000a0|  09 53 b4 99 89 06 b6 54  3c b9 f2 66 34 a8 35 20  .S.....T<..f4.5
000000b0|  b5 97 8c 1f ec 8e b9 ce  d2 9d ee 72 c4 68 0a 8c  ...........r.h..
000000c0|  8a 54 64 ed 2f c2 39 ab  15 4b f2 cb 91 29 05 fb  .Td./.9..K...)..
000000d0|  26 8b 9e 9d 4d 55 9b c8  b7 6a 4e 49 60 2c 4a 1f  &...MU...jNI`,J.
000000e0|  ef ef ed c1 f3 ab 0e 0b  c0 02 e2 6e 38 e2 6b 3b  ...........n8.k;
000000f0|  a6 ed 93 c2 89 e5 48 46  0c 3e d1 b1 c2 57 c8 e7  ......HF.>...W..
00000100|  a6 b4 c0 22 f5 a4 b4 f2  8f 9a b1 54 7a a7 0a f9  ..."......Tz...
00000110|  03 bb ce 28 f0 12 83 59  0c 7f c0 14 e7 4c bd 77  ...(...Y.....L.w
00000120|  be c9 de a2 30 9e 14 fb  d2 24 76 61 a7 53 e4 f6  ....0....$va.S..
00000130|  2b b7 00 24 2f db aa b8                           +..$/...
        0 FAIL  :  Tspi_Data_Unseal  returned (24) TPM_E_WRONGPCRVAL
unseal.c        0 FAIL  :  Tspi_Data_Unseal  returned (24) TPM_E_WRONGPCRVAL
[06/08/2022 01:26] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$
```

可以看到运行失败的错误码提示，指示错误。

4. 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。

5. 执行 ./unseal_file test.en test.de 指令，查看文件 test.de 的内容：

```
test.de
/**
 * @file test.c
 * @author WangFengwei Email: 110.visual@gmail.com
 * @brief test
 * @created 2011-06-20
 * @modified
 */

#include "common.h"

#include <stdio.h>
#include <string.h>

#include <openssl/ui.h>

#define UI_MAX_SECRET_STRING_LENGTH     256
#define UI_MAX_POPUP_STRING_LENGTH      256

int
main(int argc, char **argv)
{
        char string[UI_MAX_SECRET_STRING_LENGTH + 1];
        UINT32 strLen;
        do_ui(string, &strLen, "123\n", 0);
        printf("%s\n",string);
        return 0;
}
```

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。

6. 运行./extend -v 1.2， 再完成对 PCR 寄存器的扩展：

```
geditt: command not found
[06/08/2022 01:28] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ gedit test.de
[06/08/2022 01:28] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./extend -v 1.2
ulPcrValLen:20

Success
[06/08/2022 01:28] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$
```

7. 运行./unseal_file test.en test.de，对 test.en 文件的解密封失败：

```
[06/08/2022 01:28] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$ ./unseal_file test.en test.de
Input K1's Pin

Enter PIN:
1
        0 FAIL  :  Tspi_Data_Unseal  returned (24) TPM_E_WRONGPCRVAL
unseal_file.c   0 FAIL  :  Tspi_Data_Unseal  returned (24) TPM_E_WRONGPCRVAL
[06/08/2022 01:30] seed@ubuntu:~/trusted-computing-projectv0.3/SealUnseal$
```

4.5 秘钥迁移（KeyMigration）

进入 Key Migration 目录

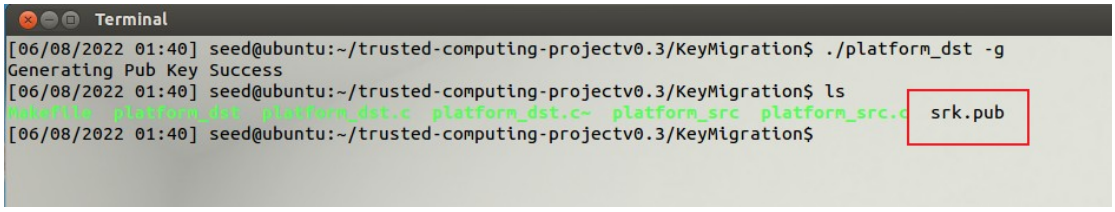
(cd /home/seed/trusted-computing-projectv0.3/KeyMigration)

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，完成编写了下图的代码：

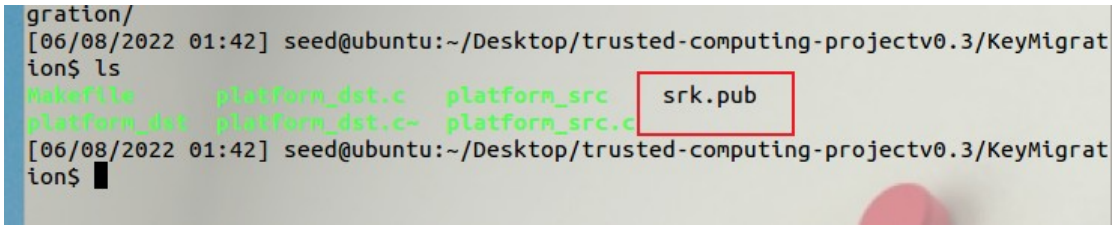
```
    // TODO Convert Migration Blob
    //  完成以下代码，参考Tspi_Key_ConvertMigrationBlob
    result = Tspi_Key_ConvertMigrationBlob(hNewMigKey,
                                           hSRK,
                                           u32RandomLen,
                                           pRandom,
                                           u32MigBlobLen,
                                           pMigBlob);
    free(pRandom);
    free(pMigBlob);
    if (TSS_SUCCESS != result) {
        print_error("Tspi_Key_ConvertMigrationBlob", result);
        Tspi_Context_Close(hContext);
        return result;
    }

    // Load the New Migrated Key
    result = Tspi_Key_LoadKey(hNewMigKey, hSRK);
    if (TSS_SUCCESS != result) {
        print_error("Tspi_Key_Load", result);
        Tspi_Context_Close(hContext);
        return result;
    }

    // verify that the new migrated key is valid
    result = sign_and_verify(hContext, hNewMigKey);
    if (TSS_SUCCESS != result) {
        print_error("sign_and_verify", result);
        Tspi_Context_Close(hContext);
        return result;
    }

    Tspi_Context_Close(hContext);

    printf("Migration Success\n");
    return 0;
```

改完代码后执行 sudo make

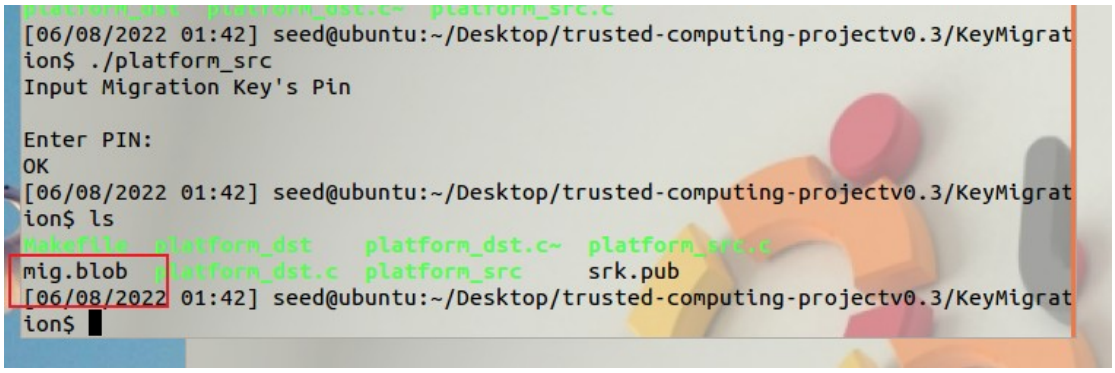
在机器 1 中运行 ./platform_dst -g 指令，回产生一个 srk.pub 的文件：



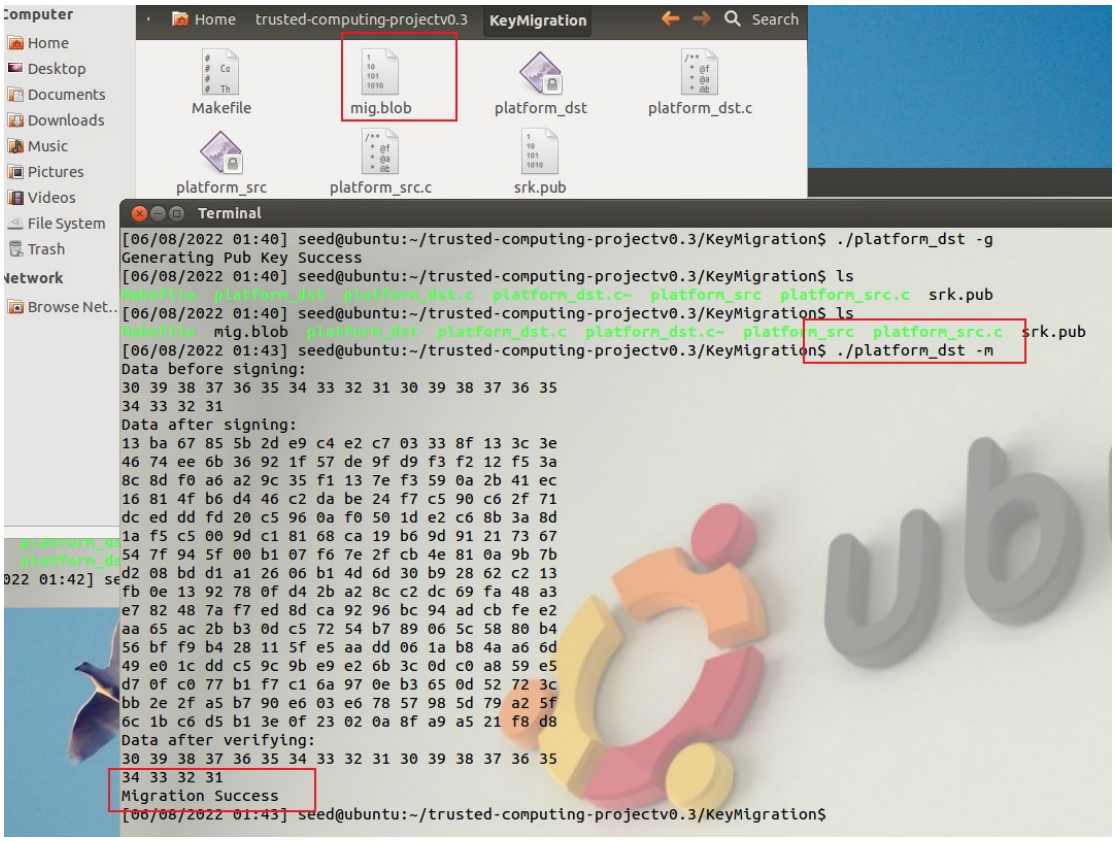
将文件 srk.pub 拷贝到机器 2 中：



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，执行./platform_dst -m



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）

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

```
Terminal
[CA:] Verify the identity binding......
[CA:]Create a fake credential and encrypt it......
[Client:]Activate the TPM identity, receiving back the decrypted credential......
Create_AIK.c:762 receive_msg_size is 44
Create_AIK.c:780        1 PASS  :  Create AIK  returned (0) TSS_SUCCESS
786 Ending remote_attestation()
801 remote_data_size 44
803 keyPubSize 256
Create_AIK.c:805 ##The content of the keyPub is:##
99 a2 ba ef 77 8b 15 1c 21 44 7d 3d ba b5 43 10
58 85 00 70 a7 29 2c e4 7f d9 86 c9 79 5e db 00
58 df c1 65 cd ce 95 bc 55 ad 08 84 20 1a 62 bc
14 f4 28 ff 31 12 1a 79 e0 67 6c 84 97 a7 a8 2f
2e 3c bb af 4e da cb ea 96 7e a8 99 ec 7f 5f bb
84 e9 5c a3 cc fc c8 e5 55 b8 23 cc 56 88 5b 79
3e 57 f3 58 ea 56 21 06 55 0b ba 3a 25 d7 d5 97
21 a7 fd d6 cc 9a 6f 36 92 d4 09 fe cb 84 70 07
c7 ea ee 5c 60 e6 e2 52 bf b0 91 8d ad a4 b3 6e
f3 cc fd be 65 05 06 7d 8e 59 24 53 56 d0 aa cb
94 b3 80 8e 83 e2 00 1f eb 6d e5 f9 6a 32 9e 84
7d a6 fa 62 0f 74 7a df 8c 18 22 1f b9 65 43 2e
c2 dd 96 20 b9 72 d5 6a 6e 32 15 01 b0 b7 76 ef
ae 20 2b f1 34 4d 00 70 7e 63 d9 44 c3 2a 53 a7
50 38 33 09 d4 92 a6 c4 5a e6 39 12 3b 00 4c c9
a5 ec 3e a1 1e 45 1a ea ee ad 65 c4 13 55 60 71
Create_AIK.c:807 ##The content of the keyPub is over##
809 credLen 104
Create_AIK.c:811 ##The content of the cred is:##
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a
Create_AIK.c:813 ##The content of the cred is over##
815 After calling remote_attestation()
817  -- remote data --
00 00 00 00 00 00 00 00 00 00 01 00 00 68 00 00 00
10 00 00 00 00 8f 7b 08 b0 9c 7b 08 b8 9d 7b 08
00 00 00 00 00 00 00 00 30 00 00 00
819  -- receive msg --
[06/08/2022 01:45] seed@ubuntu:~/trusted-computing-projectv0.3/RemoteAttestation/init$
```

3、返回上级目录

4、运行./RAServer

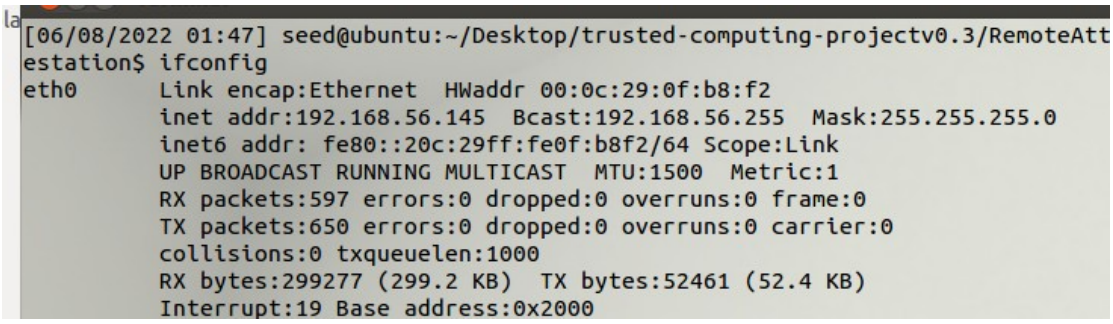
```
[06/08/2022 01:45] seed@ubuntu:~/trusted-computing-projectv0.3/RemoteAttestation/init$ cd ..
[06/08/2022 01:45] seed@ubuntu:~/trusted-computing-projectv0.3/RemoteAttestation$ ./RAServer

*********************************************************
***** Welcome to IBM Remote Attestation Server 111********
*********************************************************
```

在机器 2 中执行以下步骤:

1、进入 Remote Attestation 目录

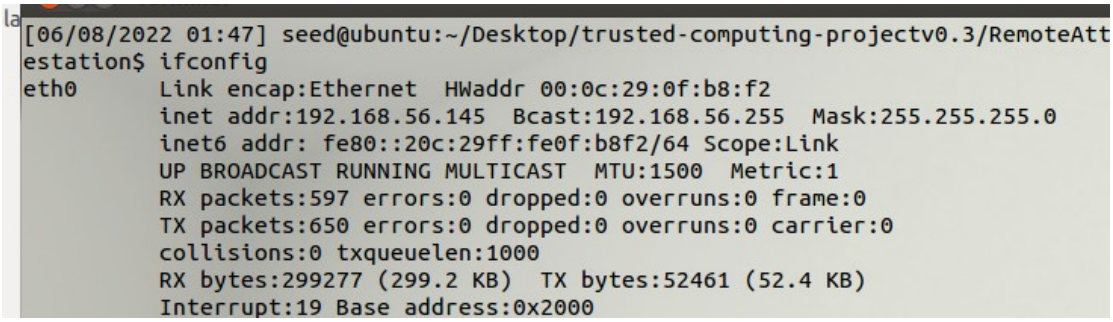
2、运行./RAClient 机器 2 的 ip 机器 1 的 ip

机器 1 的 ip：192.168.56.145



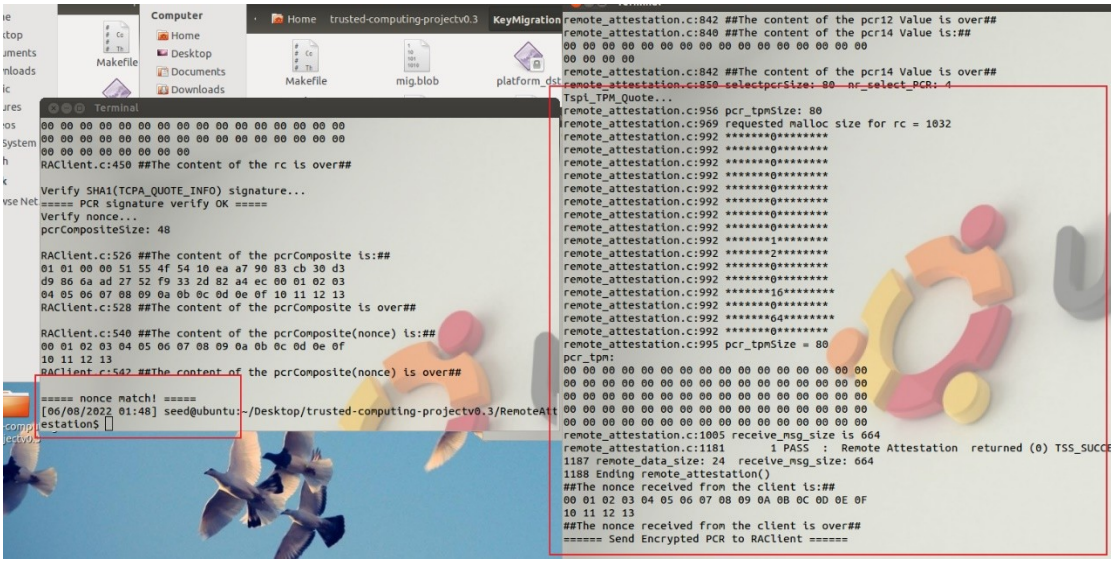
```
[06/08/2022 01:47] seed@ubuntu:~/Desktop/trusted-computing-projectv0.3/RemoteAtt
estation$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:0f:b8:f2
          inet addr:192.168.56.145  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe0f:b8f2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:597 errors:0 dropped:0 overruns:0 frame:0
          TX packets:650 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:299277 (299.2 KB)  TX bytes:52461 (52.4 KB)
          Interrupt:19 Base address:0x2000
```

机器 2 的 ip：192.168.56.145



```
[06/08/2022 01:47] seed@ubuntu:~/Desktop/trusted-computing-projectv0.3/RemoteAtt
estation$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:0f:b8:f2
          inet addr:192.168.56.145  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe0f:b8f2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:597 errors:0 dropped:0 overruns:0 frame:0
          TX packets:650 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:299277 (299.2 KB)  TX bytes:52461 (52.4 KB)
          Interrupt:19 Base address:0x2000
```

执行指令./RAClient 192.168.56.145 192.168.56.145



```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
RAClient.c:450 ##The content of the rc is over##

Verify SHA1(TCPA_QUOTE_INFO) signature...
===== PCR signature verify OK =====
Verify nonce...
pcrCompositeSize: 48

RAClient.c:526 ##The content of the pcrComposite is:##
01 01 00 00 51 55 4f 54 10 ea a7 90 83 cb 30 d3
d9 86 6a ad 27 52 f9 33 2d 82 a4 ec 00 01 02 03
04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13
RAClient.c:528 ##The content of the pcrComposite is over##

RAClient.c:540 ##The content of the pcrComposite(nonce) is:##
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13
RAClient.c:542 ##The content of the pcrComposite(nonce) is over##

===== nonce match! =====
[06/08/2022 01:48] seed@ubuntu:~/Desktop/trusted-computing-projectv0.3/RemoteAtt
estation$
```

```
remote_attestation.c:842 ##The content of the pcr12 Value is over##
remote_attestation.c:840 ##The content of the pcr14 Value is:##
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
remote_attestation.c:842 ##The content of the pcr14 Value is over##
remote_attestation.c:850 selectpcrSize: 80  nr_select_PCR: 4
Tspi_TPM_Quote...
remote_attestation.c:956 pcr_tpmSize: 80
remote_attestation.c:969 requested malloc size for rc = 1032
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******1********
remote_attestation.c:992 *******2********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******16*******
remote_attestation.c:992 *******0********
remote_attestation.c:992 *******64*******
remote_attestation.c:992 *******0********
remote_attestation.c:995 pcr_tpmSize = 80
pcr_tpm:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
remote_attestation.c:1005 receive_msg_size is 664
remote_attestation.c:1181        1 PASS  :  Remote Attestation  returned (0) TSS_SUCCE
1187 remote_data_size: 24  receive_msg_size: 664
1188 Ending remote_attestation()
##The nonce received from the client is:##
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13
##The nonce received from the client is over##
====== Send Encrypted PCR to RAClient ======
```

实现了远程证明环节。