Advanced Computer Security, EITN50

# Project: TPM

3 oktober 2017

# 1 Assignment 1: Setting up the environment

For assignment 1 the TPM1 and TSS virtual machines were used. To enable the TPM emulator to run, the following environment values in Listing 1 were set on the TPM:

**Listing 1:** Setting up parameters in the TPM

```
TPM_PATH=/home/pi/tpm/tpm4720/tpmstate
TPM_PORT=6545
```

The first command in listing 1 gives the path to the directory that stores the state of the TPM.

In order to use the TPM utility commands the environment variables in listing 2 were set.

**Listing 2:** Setting up parameters in the TPM

```
export TPM_SERVER_NAME=localhost
export TPM_SERVER_PORT=6545
```

After the required environment values were set on the TPM it was time to start the TPM emulator. This was done by opening up two terminals on the TPM machine and issue the command *tpm_server* in one of the terminal windows. In the second window the utility commands could be used.

Trousers was used on the TSS machne and this required that the enviroment variables in listing 3 were set.

**Listing 3:** Setting up parameters in the TPM

```
export TCSD_USE_TCP_DEVICE=true
export TCSD_TCP_DEVICE_PORT=6545 (default port)
export TCSD_TCP_DEVICE_HOSTNAME=10.0.2.14
```

To assure that the right variables were set on the TSS and TPM, the command *env* was entered. The *env* command returned a list of all the environment variables.

# 2 Assignment 2: Getting the TPM ready for use and creating EK keys

In order to get the TPM working to generate the EK keypair, three steps were made. Step 1: The intent with this step was to activate the the TPM.

This was done by entering the *tpmbios* command.

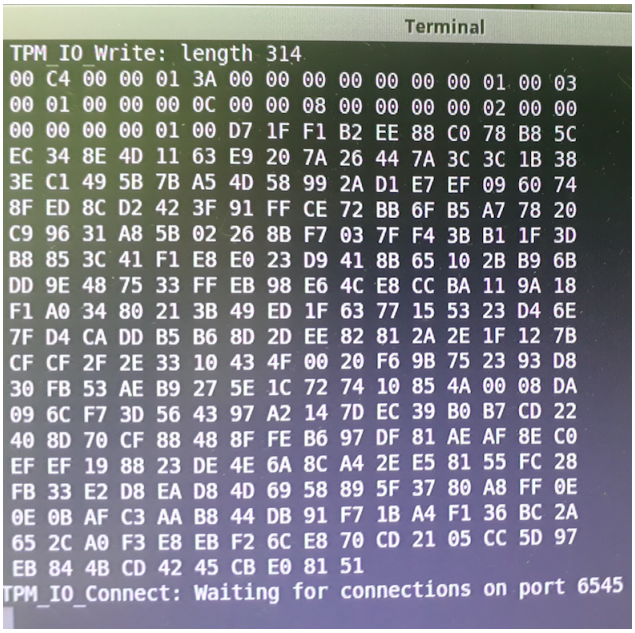
Step 2: In this step a connection to the TPM emulator on TPM1 was established by issuing the following command in listing 4.

**Listing 4:** Setting up parameters in the TPM

```
sudo -E \usr\local\sbin\tcsd -e -f
```

This command starts a deamon process on the TSS machine.

Step 3: In this step the EK key pair were created, and this was done by using the utility *createek* command. The EK key pair were then printed out on the TPM emulator terminal.




**Figur 1:** The printout of the EK keypair in the TPM emulator terminal

# 3 Assignment 3: Key hierarchy

## 3.1 Questions:

**1: The identity key is one type of signature key. Describe some differences between an identity and a signature key.**

Answer: One of the differences between the two is that the identity key is

used for attestation while the signature key is used to sign user data.

**2. Which keys can be used for file encryption?**
Answer: Legacy keys and storage keys can be used for file encryption.

**3. There is one type of key that exists, but its use is not recommended. Which key is that, and why does it exist?**
Answer: Legacy keys. The legacy keys exist because of the fact that they are compatible with older systems.

### 3.3.3 Creating a key hierarchy

| Name | Parent | Type |
|---|---|---|
| A | SRK | non migratable storage key |
| B | A | migratable storage key |
| D | B | migratable sign key |
| E | B | migratable bind key |
| F | A | non migratable sign key |
| G | A | migratable sign key |
| H | SRK | identity key |

**Figur 2**

As seen in Figure 1, the key named C could not be created. The reason for this is that a non migratable key can not be a child of a migratable key.

# 4 Assignment 4

### 3.4.4 Questions:

**1. Is it possible for a migratable key to be the parent of a non-migratable key?**
Answer:No, it is not possible for a migratable key to be the parent of non migratable keys. When a key is migrated, all of it´s children are migrated aswell. Thus, all children of a migratable key have to be migratable.

**2. Which command is the first to be executed when performing a key migration?**
Answer: The *TPM_CreateMigrationBlob* is the command that implements the first step in the process of moving a key to a new parent or platform.

**3. Give a short description of the command `TPM_ConvertMigrationBLob`.**
Answer: This command takes a migration blob and creates a normal wrapped blob. The migrated blob must be loaded into the TPM using the normal *TPM_ loadKey* function.

**4. Which TPM command loads the migrated keys into the TPM?**
Answer: *TPM_loadkey*

**5. Is it the TPM or the TSS that handles the transfer of the migration blob?**
Answer: The TSS handles the transfer of the migration blob.

## Key migration in the TPM emulator

### 3.4.4 Questions:

**1. Do the above migration and document in your report.** In TPM1 the following command was issued to create a migration blob:

```
migrate -hp < parent key Handle> -pwdo ooo -im A.key -pwdm <migration pa
```

Then TPM2 was connected and the following command was issued, in TPM2, to load the migrationblob:

```
loadmigrationblob -hp <TPM2 Storage key handle> -pwdp sss -if migrationb
```

All the keys, except key C were migrated successfully. Key C was not able able to be created as mentioned in assignment 3.3.3. Keys D and E are children of key B and were therefore migrated to TPM2 when B was migrated.

**2. There are other ways to migrate keys. When do you use a key of type `TPM_KEY_USAGE` =`TPM_Migrate`**

Answer: When using migration authority.

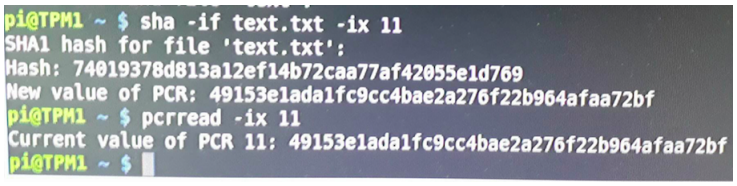
**3. What is the rewrap option of the migrate command used for?**
Answer: The rewrap option is used to directly move a key to a new parent.

# 5 Assignment 5: Extending values to PCRs

## Questions

**1. Describe one TPM command that can be used to extend a SHA-1 digest to a PCR.**
Answer: The `TPM Extend` command can be used to extend a SHA-1 digest to a PCR by adding a new measurment to a PCR.

**2. Describe which TPM command that can be used to read a PCR value.**
Answer: The `TPM PCRread` command can be used to read a PCR value. The command provides non-cryptographic reporting of the contents of a named PCR.




**Figur 3:** The new PCR value

# 6 Assignment 6: File encryption

## 3.6.1 Questions:

**1. Why is TSS_Bind a TSS command, and not a TPM command?**
Answer: The encryption is done outside of the TPM and is therefore not a TPM command.

**2. Give some differences between Data binding and Data sealing**
Answer: Data binding uses a symmetric key for encryption and data sealing uses asymmetric encryption.

**3. Can a key used for data sealing be migrated to another TPM?**
Answer: No, because sealing data to one TPM platform makes it illegal to migrate a key to another TPM.

### 3.6.3 Instructions: Data Sealing using the TPM emulator

**Why doesnt the key have to be loaded inside the TPM when encrypting, but it has to be when decrypting?**
Answer: When encrypting only the public key is needed while decryption uses the private key, which is not accessible outside of the TPM.

**Now migrate the binding key to TPM2 and see if you can decrypt the file there too. Explain what you observe.**

To be able to decrypt the binding key is needed. Since the binding key was migrated to TPM2, the file was able to be decrypted there too.

**Test if you can do a sealing with a legacy key, a binding key or a signing key. If not, why?**
It was not possible to do a sealing with a legacy key, binding key or signing key. The reason for this is that sealing only can be done with storage keys.

**Now migrate the storage key to TPM2 and see if you can unseal the file there too. Explain what you observe.** The storage key is not migratable and could therefore not be migrated to TPM2. Since TPM2 did not have access to the storage key, unsealing was not achieved.

# 7 Assignment 7

**1. In the above, could the verifyfile command have been done by another TPM?**
Answer: Yes, this is possible because only the public key is used.
**2. Which TPM command is used to decrypt the file?**
Answer: The command that is used to decrypt files is TPM_UnBind.
**3. Can the decryption based authentication be done by using data sealing instead of binding?**
Answer:No it can not because the TPMs have different PCR values.

### 3.7.3

**1. Sign a file with some text in it by loading a signature key into the TPM1 and use this key to sign the file using the utility signfile. Let TPM2 verify the signature by using the utility verifyfile.**

```
signfile -hk <key handle> -if text -os signedText
verifyfile -is signedText -if text -ik Sign.pem
```

Verification was successfull.

**2. Encrypt a file by creating a binding key and load it into the TPM and then encrypt a text file using the command bindfile Then decrypt it using the command unbindfile.**

```
bindfile -ik Bind.pem -if text -of enctext
unbindfile -hk <key handle> -if enctext -of dcr
```

Also successfull.

# 8 Assignment 8: Attestation

**Create an AIK (Attestation Identity Key) using the command identity. Use it to quote a PCR value, like the PCR with the hash digest of tpmbios. The verification of the quote will be done automatically**

```
identity -pwdo ooo -la label -pwds sss -ok Identity

loadkey -ik Identity.key -hp40000000 -pwdp sss

/home/tss/tpm/tpm4720/libtpm/utils/quote -v
-hk <key handle> -bm <pcr hash digest>
```

Successfull.

## Decrypt based attestation

Creating key:

```
createkey -hp <parent key handle> -kt e -ix 11 <digest> -ok dkey
```

Sealing:

```
sealfile -hk <Key handle> -if text1 -of text1out
```

Unsealing:

```
unsealfile -hk <key handle> -if text1out -of out1
```

After the text file was changed an error was found.

# 9 Assignment 9: Creating TPM program

The program uses Tspi_TPM_GetRandom to generate 8 random numbers.
The source code can be found in Appendix A.

# 10 Appendix A: Source code

```
// INSERT TPM COMMANDS HERE:
BYTE bytes[8];
int size=8;

result=Tspi_TPM_GetRandom(hTPM, 8, bytes);
int i;
for (i=0;i<8; i++)
printf("%2x\n",bytes[i])
// END OF APP
// Free memory
result = Tspi Context FreeMemory ( hContext , NULL ) ;
```