Lab1 - Trusted Platform Module

Welcome to the Trusted Computing Lab, in this lab you will learn about Trusted Platform Modules (TPMs) and how to use them. In this lab, there are few tasks, which are designed as walkthroughs. The goal is to push your knowledge and use what you have learned to solve problems and issues when working with the TPM. TPMs are used in most computing platforms (Servers, Desktops, ..). This laboratory will explore some of the key concepts surrounding trusted computing and key management.

Joining Discord

Many cyber security modules are being run on Discord instead of Microsoft Teams, which allows demonstrators and lecturers a number of tools to engage with students through text and voice chat.

What is Discord? it's like teams except it has superior audio and video quality, also has a lot of features that make your learning experience much better. Want to know more? read this article link.

In order to join the laboratory support discord, you will need to sign in and associate a discord account with your university account.

1. First, you will need to join the University VPN. To do this, go to the following link to download and install the client for your OS here. Make sure you are connected via VPN before you continue.
2. Make a Discord account at https://discord.com/, or make sure you are signed into your existing account either on the native app or in the web browser.
3. Login to https://discord.ecs.soton.ac.uk/ with your university login.
4. Press the lab link to join the virtual lab, and accept the discord invite.

Experimental Setup

For the laboratory sessions in this module we will be making use of tools.

Vagrant, is a tool which allows virtual machines with specific configurations to be spun up from configuration files. In order to use it, you will need to download some software. This tool will ensure that you have a clean experimental setup every time. It will also help you get up and ready for the lab in no time, all the tools and data will be inside the vagrant script. All you have to do is run it and enjoy.

Virtual Box, is a tool that allows you to run virtual machines on your computer and its a pre-requisite for Vagrant.

Git, is a tool that help you do what we call source control. You will be using this to download the new lab every week.

Please download and install them in this order:

1. Download Virtualbox from https://www.virtualbox.org/wiki/downloads.
2. Download Vagrant from https://www.vagrantup.com/downloads.
3. Download git (Windows) from https://git-scm.com/downloads. If you're running mac or linux, you should already have this installed.
4. When installing git, make sure you tick the option to "Add to path", which will allow you to run git from the terminal.

Once both of these packages have been installed, you should be able to open up a terminal on your computer (cmd on Windows, Terminal on Mac/Linux) and type vagrant to check it is installed and working.

Listing 1: Verify that vagrant is installed

```
#!/bin/bash
vagrant -v
```

If you get an output showing all supported vagrant commands, you are good to move on to the next step.

Your First Vagrant Machine

Vagrant works by reading configuration files in your current directory and using them to create VMs. Hence, in order to spin up the VM, you may need to use the cd command in your terminal to change-directory into the correct folder. Change directory into a working folder from which you wish to complete the labs. **You will need to make it if it doesn't exist, and then cd into it for example:**

```
#!/bin/bash
cd Documents/comp3217-labs
```

Download the lab1 VM image from the UoS Git server by typing the following:

```
#!/bin/bash
git clone https://git.soton.ac.uk/comp3217/trusted_computing.git
```

Change into the lab1 folder so we can use vagrant:

```
#!/bin/bash
cd trusted_computing
```

You should now be able to run the following command to

```
#!/bin/bash
vagrant up
```

If you have the virtualbox window open, you will notice that a new VM appears, and vagrant begins to build it. It should take a couple of minutes.

When the build process is complete, you can connect to a shell on your newly created VM using the following command:

```
#!/bin/bash
vagrant ssh
```

Submission Instructions

Please Submit your solution report (individually) in PDF format + Code all packed in a .zip file to this link `https://handin.ecs.soton.ac.uk`.

Deadline

The assignment deadline is on 11/03/2022

Contents

1 Setting up the environment **4**
1.1 Prerequisites 4

2 TPM2 Simulator **4**
2.1 Installation 4
2.2 Run TPM simulator 4

3 Getting Familiar with TPM **4**

4 Part-1: Cryptography Basics (20%) **5**

5 Part-2 - Policy (20%) **6**
5.1 Example of creating an OR policy 6
5.2 Question 7

6 Part-3 Quote (20%) **7**

7 Part-4 Putting It All Together (40%) **8**

1 Setting up the environment

In this lab we will work with a software TPM and connect to the TPM simulator via sockets (port=3231). You will use '-T mssim:host=localhost,port=2321' in each TPM command that indicates using TPM simulator. An alternative option is to define the TPM2TOOLS_TCTI variable as indicated below. Using the tpm-tools we will examine the commands learned in the lecture and how to apply them to guarantee confidentiality of keys and prove platform identity to a remote attester.

1.1 Prerequisites

- Vmware Fusion for launching Virtual machine
- Ubuntu 20.04.1 LTS `https://ubuntu.com/download/desktop`
- tpm-tools

2 TPM2 Simulator

After booting your virtual machine using Vmware Fusion (or Virtual Box) you will need to install the TPM2 simulator and run it in a separate terminal tab.

2.1 Installation

```
sudo apt update
sudo apt install snapd tpm2-tools
sudo snap install tpm2-simulator-chrisccoulson --edge
```

2.2 Run TPM simulator

Open the Terminal and run the following command in one of the tabs to start the TPM:

```
comp3217@ubuntu:~$ tpm2-simulator-chrisccoulson.tpm2-simulator
```

Expected output is the TPM waiting for connection when calling TPM commands

```
LIBRARY_COMPATIBILITY_CHECK is ON
TPM command server listening on port 2321
Platform server listening on port 2322
```

3 Getting Familiar with TPM

Clearing the TPM: Clearing is the first thing you need to do when working with a new TPM. Following that you will take ownership of the TPM which guarantees you are starting from new state with new set of keys.

```
tpm2_startup -c -T mssim:host=localhost,port=2321
```

Reading the Platform Configuration Registers (PCR): Reading PCR Sha256 banks

```
tpm2_pcrread -T mssim:host=localhost,port=2321 sha256
```

Output:

```
sha256:
0 : 0x0000000000000000000000000000000000000000000000000000000000000000
1 : 0x0000000000000000000000000000000000000000000000000000000000000000
2 : 0x0000000000000000000000000000000000000000000000000000000000000000
3 : 0x0000000000000000000000000000000000000000000000000000000000000000
4 : 0x0000000000000000000000000000000000000000000000000000000000000000
5 : 0x0000000000000000000000000000000000000000000000000000000000000000
6 : 0x0000000000000000000000000000000000000000000000000000000000000000
7 : 0x0000000000000000000000000000000000000000000000000000000000000000
8 : 0x0000000000000000000000000000000000000000000000000000000000000000
```

```
9 : 0x0000000000000000000000000000000000000000000000000000000000000000
10: 0x0000000000000000000000000000000000000000000000000000000000000000
11: 0x0000000000000000000000000000000000000000000000000000000000000000
12: 0x0000000000000000000000000000000000000000000000000000000000000000
13: 0x0000000000000000000000000000000000000000000000000000000000000000
14: 0x0000000000000000000000000000000000000000000000000000000000000000
15: 0x0000000000000000000000000000000000000000000000000000000000000000
16: 0x0000000000000000000000000000000000000000000000000000000000000000
17: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
18: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
19: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
20: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
21: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
22: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
23: 0x0000000000000000000000000000000000000000000000000000000000000000
```

Generating Random Number Example reading 15 bytes

```
tpm2_getrandom -T mssim:host=localhost,port=2321 --hex -f 15
```

Creating and signing with an RSA key

```
export TPM2TOOLS_TCTI=mssim:host=localhost,port=2321
// creating a key
tpm2_createprimary -C e -c primary.ctx

tpm2_create -G rsa -u rsa.pub -r rsa.priv -a 'fixedtpm|fixedparent|
    ↪ sensitivedataorigin|userwithauth|restricted|decrypt' -C primary.ctx

tpm2_load -C primary.ctx -u rsa.pub -r rsa.priv -c rsa.ctx
```

```
echo "data_to_sign" > message.dat
sha256sum dummy.txt | awk '{ print "000000 " $1 }' | xxd -r -c 32 > data.in.
    ↪ digest
```

```
//signing with a key
tpm2_sign -c rsa.ctx -g sha256 -o sig.rssa message.dat

//using the key will fail because the policy allow using the key for decrypt

// repeat the steps in this listing with this command instead tpm2_create -G rsa
    ↪  -u rsa.pub -r rsa.priv -C primary.ctx

//signing with a key
tpm2_sign -c rsa.ctx -g sha256 -o sig.rssa message.dat

tpm2_verifysignature -c rsa.ctx -g sha256 -s sig.rssa -m message.dat
```

You can find information about the TPM commands on:
`http://manpages.ubuntu.com/manpages/bionic/man8/tpm2_startup.8.html`
`https://trustedcomputinggroup.org/resource/tpm-library-specification/`

4 Part-1: Cryptography Basics (20%)

1. Download dummy.txt, hash it using sha256 and extend the hash to PCR-10. What are the values of the PCRs before and after? Explain the result of PCR-10 following extending the hash value.

2. Create a new EK and perform signing followed by verifying operations. How do you guarantee that only an admin can use the EK?
3. Create a new SRK, specify the authorization "value = 9854" to prevent another user from using the new generated SRK. Attempt to perform a bruteforce attack starting from 0000 to 9999. Are you able to launch a successful attack? Explain the result you get from the TPM.
4. Create a new symmetric key (AES-128) with a policy of your choice. What is the advantage of using a TPM to generate the key vs generating a symmetric key with OpenSSL?

5 Part-2 - Policy (20%)

In this part you are going to work with policy and use specific operations only when this policy is satisfied. In Part-1 we have worked without taking ownership. To take ownership we need to execute the following:

```
%%clearing the tpm from previous ownership
tpm2_clear
tpm2_startup -c
tpm2_changeauth -c owner password
tpm2_changeauth -c endorsement password
```

The command we used above wont work anymore without the right authorization, this will guarantee that no other user can generate primary keys. We will need to add "-P password" .

5.1 Example of creating an OR policy

PCR1 policy

```
tpm2_startauthsession -S session.ctx
tpm2_policypcr -S session.ctx -l sha1:23 -L set1.pcr0.policy
tpm2_flushcontext session.ctx
rm session.ctx
```

PCR2 policy

```
tpm2_pcrextend 23:sha1=f1d2d2f924e986ac86fdf7b36c94bcdf32beec15
tpm2_startauthsession -S session.ctx
tpm2_policypcr -S session.ctx -l sha1:23 -L set2.pcr0.policy
tpm2_flushcontext session.ctx
rm session.ctx
```

Create a policyOR resulting from compounding the two unique pcr policies in an OR fashion

```
tpm2_startauthsession -S session.ctx
tpm2_policyor -S session.ctx -L policyOR -l sha256:set1.pcr0.policy,set2.pcr0.policy
tpm2_flushcontext session.ctx
rm session.ctx
```

Create a sealing object with auth policyOR created above.

```
   tpm2_createprimary -C o -P password -c prim.ctx
   tpm2_create -g sha256 -u sealkey.pub -r sealkey.priv -L policyOR -C prim.ctx -i-
       ↪  <<< "secretpass"
   tpm2_load -C prim.ctx -c sealkey.ctx -u sealkey.pub -r sealkey.priv
```

Attempt unsealing by satisfying the policyOR by satisfying SECOND of the two poli- cies.

```
tpm2_startauthsession -S session.ctx --policy-session
tpm2_policypcr -S session.ctx -l sha1:23
tpm2_policyor -S session.ctx -L policyOR -l sha256:set1.pcr0.policy,set2.pcr0.policy
unsealed=‘tpm2_unseal -p session:session.ctx -c sealkey.ctx‘
```

```
echo $unsealed
tpm2_flushcontext session.ctx
rm session.ctx
```

Extend the pcr to emulate tampering of the system software and hence the pcr val- ue.

```
    tpm2_pcrextend 23:sha1=f1d2d2f924e986ac86fdf7b36c94bcdf32beec15
```

Attempt unsealing by trying to satisfy the policOR by attempting to satisfy one of the two policies.

```
    tpm2_startauthsession -S session.ctx --policy-session
    tpm2_policypcr -S session.ctx -l sha1:23
```

This should fail:

```
tpm2_policyor -S session.ctx -L policyOR -l sha256:set1.pcr0.policy,set2.pcr0.policy
tpm2_flushcontext session.ctx
rm session.ctx
```

Reset pcr to get back to the first set of pcr value

```
tpm2_pcrreset 23
```

Attempt unsealing by satisfying the policyOR by satisfying FIRST of the two poli- cies.

```
tpm2_startauthsession -S session.ctx --policy-session
tpm2_policypcr -S session.ctx -l sha1:23
tpm2_policyor -S session.ctx -L policyOR -l sha256:set1.pcr0.policy,set2.pcr0.policy
unsealed=‘tpm2_unseal -p session:session.ctx -c sealkey.ctx‘
echo $unsealed
tpm2_flushcontext session.ctx
rm session.ctx
```

5.2 Question

1. Tom would like to store a secret file on his hardrive. He plans to use his Laptop TPM to securely store his file encrypted. Design and implement a solutions using TPM and policy that would allow Tom to protect the secrets and seal it to his laptop only. *Note: There are multiple ways to choose the authorisation method to access the key/data*

6 Part-3 Quote (20%)

In this part you are going to work with Quote. tpm2_quote provide quote and signature for given list of PCRs in given algorithm/banks. PCRs are used to store the identity and configuration of the platform and tpm2_quote use a primary key from the endorsement hierarchy.

Try performing the next sequence:

```
 tpm2_createprimary -C e -c primary.ctx
 tpm2_create -C primary.ctx -u key.pub -r key.priv
 tpm2_load -C primary.ctx -u key.pub -r key.priv -c key.ctx
 tpm2_quote -Q -c key.ctx -l 0x0004:16,17,18+0x000b:16,17,18d
```

In this task we will help your bank in using the TPM to enhance the ability of it's users in making trust decision about the servers used in the communication between the user and the bank. In this part we would like to use quotes and the identity provided by the TPM to make a trust decision about the platforms used by the bank on the communication.

- Can we use the SRK to produce a quote? Why is the endorsement hierarchy used for quote?

- You want your bank to provide a quote of the PCRs on a server before you start sharing private information. Assume that the public key to verify a quote can be shared with you in a secure manner.Your task is to demonstrate how your bank would generate a quote to prove the state of their server/software. ; either a detailed description or an example implementation. You are free to use placeholder measurements for the PCR values used in this remote attestation process. If you use any placeholder values, ensure you explain what would take their place in a real environment. The purpose of providing PCR is for you to make a decision if you can trust your bank server. What information do you want to request from your bank to capture in the PCRs? Think about why this information is important for you to make a trust decision.
- Describe the protocol between you and your bank to request, exchange and verify the quote. How can you be sure of the identity/code/configuration of the bank server and if the quote is legitimate?

Note: If we think the bank server is compromised and some kind of an attacker is sending the same (old) quote all over again to you, in order to make you believe nothing is wrong with the server. Can you provide an algorithm/method to prevent this kind of attack from happening again?

7 Part-4 Putting It All Together (40%)

Your task is to move data securely from server A to server B using keys stored within the TPM. As learned in class, TPMs provide identity of the device and can verify/attest of the software that is running on the system. In this question, assume that Server A and Server B has exchanged the keys needed to allow identifying each other. In this question you are asked to:

- Define the protocol between server A and Server B and use TPM commands to help you implement secure communication. You will need to:
  1. Define and implement the key migration of the symmetric key from server A to server B
  2. Define and implement the key migration of the asymmetric key from server A to server B
  3. Decrypt Symmetric key
  4. Verify using the signatures needed from (1) and (2)

Note: Your task is not to implement the communication part (e.g. TLS between A & B). Your task is to implement and describe when the TPM is being used and for what purpose. How do you guarantee that a specific key is originating and protected by a TPM on Server A?