vTPM configuration

My environment

- OS: Ubuntu 20.04 minimum installation

Dependencies

```
sudo apt update && \
sudo apt install -y make \
  htop \
  git \
  vim \
  python3 \
  python3-pip \
  python3-venv \
  python3-dev && \
  sudo snap install --classic code
```

VNC Installation

Install dependencies:

```
sudo apt install -y tigervnc-standalone-server tigervnc-xorg-extension
```

Setup password `vncpasswd`

Configure desktop environment

```
vim $HOME/.vnc/xstartup
```

and add

```
#!/bin/sh

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
vncconfig -iconic &
dbus-launch --exit-with-session gnome-session &
```

Give exec permission

```
sudo chmod +x $HOME/.vnc/xstartup
```

Start VNC server

```
vncserver -localhost no -geometry 1920x1080 -depth 24
```

To kill VNC server

```
vncserver -kill :<session-number>
```

TPM-simulator

Install the necessary dependencies of the TPM-simulator, build and execute as a daemon service.

Setup

```
sudo apt install -y lcov \
  pandoc \
  autoconf-archive \
  liburiparser-dev \
  libdbus-1-dev \
  libglib2.0-dev \
  dbus-x11 \
  libssl-dev \
  autoconf \
  automake \
  libtool \
  pkg-config \
  gcc \
  build-essential \
  libcurl4-gnutls-dev \
  libgcrypt20-dev \
  libcmocka-dev \
  uthash-dev \
  doxygen \
  curl \
  acl \
  libjson-c-dev \
  libusb-1.0-0-dev
```

```
# optional
sudo apt install -y --allow-downgrades \
    libc6=2.31-0ubuntu9.7 \
    libc-bin=2.31-0ubuntu9.7
```

Download TPM simulator

```
wget https://jaist.dl.sourceforge.net/project/ibmswtpm2/ibmtpm1661.tar.gz
```

Create installation directory to extract towards into

```
mkdir ibmtpm1661 && \
cd ibmtpm1661 && \
tar -xzvf ../ibmtpm1661.tar.gz
```

Enter src/ directory and execute build

```
cd src/ && \
sudo make -j$(nproc)
```

Copy the built executable to your bin directory

```
sudo cp tpm_server /usr/local/bin
```

Configure TPM simulator as a daemon service in ubuntu

```
sudo vim /lib/systemd/system/tpm-server.service
```

Add the following content to the file

```
[Unit]
Description=TPM2.0 Simulator Server daemon
Before=tpm2-abrmd.service
[Service]
ExecStart=/usr/local/bin/tpm_server
Restart=always
Environment=PATH=/usr/bin:/usr/local/bin
[Install]
WantedBy=multi-user.target
```

Reload daemon and start the service

```
sudo systemctl daemon-reload && \
sudo systemctl start tpm-server.service && \
sudo systemctl enable tpm-server.service && \
sudo systemctl status tpm-server.service
```

tpm2-tss

To be able to interface with a TPM (simulator or not), we would need to install tpm2-tss. Tpm2-tss is an opensource library that implements Trusted Computing Group's (TCG) TPM2 Software Stack (TSS). Details and the APIs that implements can be found in its official github repository.

Setup

Install json-c on top of the other dependencies we've installed from the previous step

```
sudo apt-get install -y libjson-c-dev
```

Download release 4.0.1 of tpm2-tss

```
wget https://github.com/tpm2-software/tpm2-
tss/releases/download/4.0.1/tpm2-tss-4.0.1.tar.gz
```

Extract

```
tar -xzvf tpm2-tss-4.0.1.tar.gz && \
cd tpm2-tss-4.0.1/
```

Configure

```
./configure --disable-doxygen-doc
```

Install

```
sudo make install
```

Update the run-time bindings before executing a program that links against the libraries, as prep for the next item that we will install

```
sudo ldconfig
```

tpm2-abrmd

Normally with a real physical TPM device, there's the resource manager (tpmrm) that… well let's just quote what trusted computing group defines it:

> manages the TPM context in a manner similar to a virtual memory manager. it swaps objects, sessions, and sequences in and out of the limited TPM memory as needed. This layer is mostly transparent to the upper layers of the TSS and is not mandatory. However, if not implemented, the upper layers will be responsible for TPM context management

In our case, since this is a software TPM we would have to install an access broker implementation of it, which is what tpm2-abrmd (TPM2 Access Broker & Resource Manager) is all about.

Setup

Add tss user and group as that will be the one used by the package during bus name claim normally tpm2-tss installation will do this for you, but in case none yet then this is the command to do it

```
sudo useradd --system --user-group tss
```

Add user to new group

```
sudo usermod -aG tss dj-d
```

Run to not reboot the system

```
newgrp tss
```

Download tpm2-abrmd

```
wget https://github.com/tpm2-software/tpm2-
abrmd/releases/download/2.3.1/tpm2-abrmd-2.3.1.tar.gz
```

Extract, configure, and install

```
tar -xzvf tpm2-abrmd-2.3.1.tar.gz && \
cd tpm2-abrmd-2.3.1
```

```
sudo ldconfig
```

`--with-dbuspolicydir` this is th directory where a policy that will allow tss user account to claim a name on the D-Bus system bus. `--with-systemdsystemunitdir`: systemd unit directory that is needed for tpm2-abrmd daemon to be started as part of the boot process of your unit.

```
./configure --with-dbuspolicydir=/etc/dbus-1/system.d --with-
systemdsystemunitdir=/usr/lib/systemd/system
```

```
sudo make install
```

Add tpm2-abrmd into the system service during the build+install phase, a sample service definition is placed under `/usr/local/share/dbus-1/system-services/` copy it to the system-services directory

```
sudo cp /usr/local/share/dbus-1/system-
services/com.intel.tss2.Tabrmd.service /usr/share/dbus-1/system-services/
```

Restart DBUS

```
sudo pkill -HUP dbus-daemon
```

Modify the tpm2-abrmd service configuration. There specify that the TCTI interface to be used is that of the software TPM this will make it act as the access broker for the software TPM

```
sudo rm /lib/systemd/system/tpm2-abrmd.service && \
sudo vim /lib/systemd/system/tpm2-abrmd.service
```

Content of tpm2-abrmd.service then would be:

```
[Unit]
Descript=TPM2 Access Broker and Resource Management Daemon

[Service]
Type=dbus
Restart=always
RestartSec=5
BusName=com.intel.tss2.Tabrmd
StandardOutput=syslog
ExecStart=/usr/local/sbin/tpm2-abrmd --tcti="libtss2-tcti-
mssim.so.0:host=127.0.0.1,port=2321"
User=tss
```

```
[Install]
WantedBy=multi-user.target
```

Run the service and check its state. Should be in active state if all is good.

```
sudo systemctl daemon-reload && \
sudo systemctl start tpm2-abrmd.service && \
sudo systemctl enable tpm2-abrmd.service && \
sudo systemctl status tpm2-abrmd.service
```

tpm2-tss-engine

To be able to use openssl with tpm, we would to install tpm2-tss-engine.

Setup

Get version compatible, for this article's example its 1.2.0

```
wget https://github.com/tpm2-software/tpm2-tss-
engine/releases/download/1.2.0/tpm2-tss-engine-1.2.0.tar.gz
```

Extract

```
tar -xzvf tpm2-tss-engine-1.2.0.tar.gz && \
cd tpm2-tss-engine-1.2.0/
```

Configure

```
./configure
```

Install

```
sudo make install
```

```
sudo ldconfig
```

tpm2-tools

To be able to tinker with TPM itself (i.e. checking how to read PCRs, creating private key under a heirarchy, etc), tpm2-tools provides the commands to help with this.

Download from official release

```
wget https://github.com/tpm2-software/tpm2-
tools/releases/download/5.5/tpm2-tools-5.5.tar.gz
```

Extract

```
tar -xzvf tpm2-tools-5.5.tar.gz && \
cd tpm2-tools-5.5/
```

Configure

```
./configure
```

Install

```
sudo make install
```

Test!

```
tpm2_pcrread
```

Setup env variables and FAPI profile

Edit file `P_RSA2048SHA256.json`

```
sudo vim /usr/local/etc/tpm2-tss/fapi-profiles/P_RSA2048SHA256.json
```

and remove `0x81000001` from `srk_template`

```
# before
{
  ...
  "skr_template": "system,restricted,decrypt,0x81000001"
  ...
}
```

```
# after
{
  ...
  "skr_template": "system,restricted,decrypt"
  ...
}
```

create a copy of `fapi-config.json` that works with RSA:

```
sudo cp /usr/local/etc/tpm2-tss/fapi-config.json /usr/local/etc/tpm2-
tss/fapi-config-rsa.json
```

After that edit file `fapi-config-rsa.json`

```
sudo vim /usr/local/etc/tpm2-tss/fapi-config-rsa.json
```

and replace `P_ECC256SHA256` with `P_RSA2048SHA256` and add option `"ek_cert_less": "yes"`

```
# before
{
    "profile_name": "P_ECC256SHA256",
    ...
}

# after
{
    "profile_name": "P_RSA2048SHA256",
    ...,
  "ek_cert_less": "yes"
}
```

Add env var `TSS2_FAPI` to your `.bashrc` (with this you specify what scheme use for encryption)

```
echo "export TSS2_FAPICONF=/usr/local/etc/tpm2-tss/fapi-config-rsa.json" >>
$HOME/.bashrc && \
source $HOME/.bashrc
```

Check that the path key is `P_RSA2048SHA256`

```
tss2_list
```

GitHub repo

Clone repository

```
git clone https://github.com/dj-d/tpm-test.git
```

Token

```
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
```

Config

```
cd tpm-test/ && \
python3 -m venv venv && \
source venv/bin/activate && \
pip install tpm2-pytss && \
code .
```

Reference

Medium tutorial Uncompleted function "verify_signature" ECC Encryption not yet supported How to change encryption scheme from ECC to RSA?