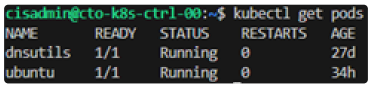
Signing and Verifying Container Images with TPM Assurance

TPMs are found in many devices and can offer the ability to detect drift while at the same time verifying signatures. This document shows how to sign and verify container image signatures with TPM assurance at runtime.

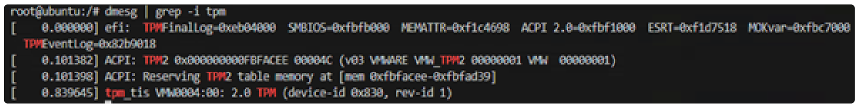
1. Starting from the control plane command line, list pods with “kubectl get pods”.
2. The “ubuntu” pod is running a container with an ubuntu image.



```
cisadmin@cto-k8s-ctrl-00:~$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
dnsutils   1/1     Running   0          27d
ubuntu     1/1     Running   0          34h
```

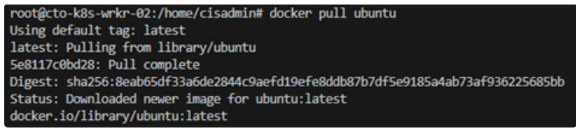
The spec is based on a Kubernetes debug pod that runs an Ubuntu image. This spec must be used or the will just exit immediately. Directions are here.

3. Run “kubectl exec -it ubuntu bash” to get a shell into the running container (Get a Shell to a Running Container).
4. Verify TPM root of trust is present and available. Run “dmesg | grep -i tpm”. The screenshot confirms the presence of a VMware vTPM.



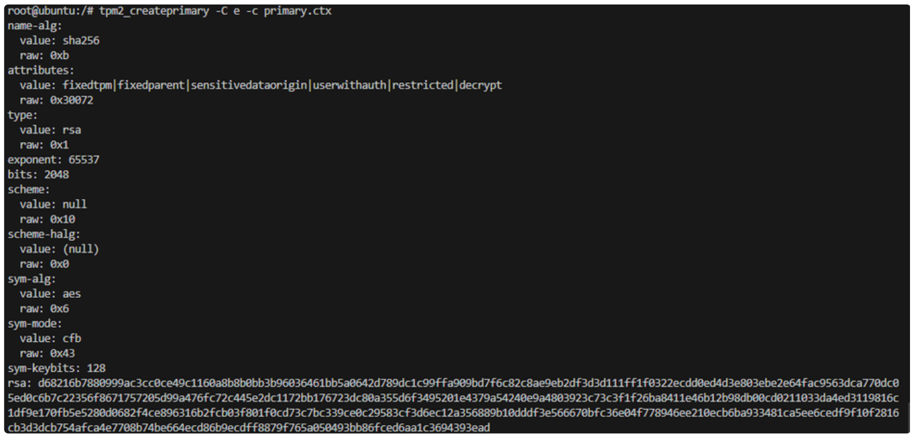
```
root@ubuntu:/# dmesg | grep -i tpm
[    0.000000] efi:  TPMFinalLog=0xeb04000  SMBIOS=0xfbfb000  MEMATTR=0xf1c4698  ACPI 2.0=0xfbf1000  ESRT=0xf1d7518  MOKvar=0xfbc7000
  TPMEventLog=0x82b9018
[    0.101382] ACPI: TPM2 0x00000000FBFACEE 00004C (v03 VMWARE VMW_TPM2 00000001 VMW  00000001)
[    0.101398] ACPI: Reserving TPM2 table memory at [mem 0xfbfacee-0xfbfad39]
[    0.839645] tpm_tis VMW0004:00: 2.0 TPM (device-id 0x830, rev-id 1)
```

5. Install TPM2-Tools by running “apt-get install tpm2-tools”. TPM2-tools are needed to create the key to be used to sign the container image. This key captures the PCR values that if changed would cause a different key pair value to be created.
6. Kubernetes often uses docker as an image registry. Run “docker pull ubuntu” to download the image file for Ubuntu. Install docker if not already installed.



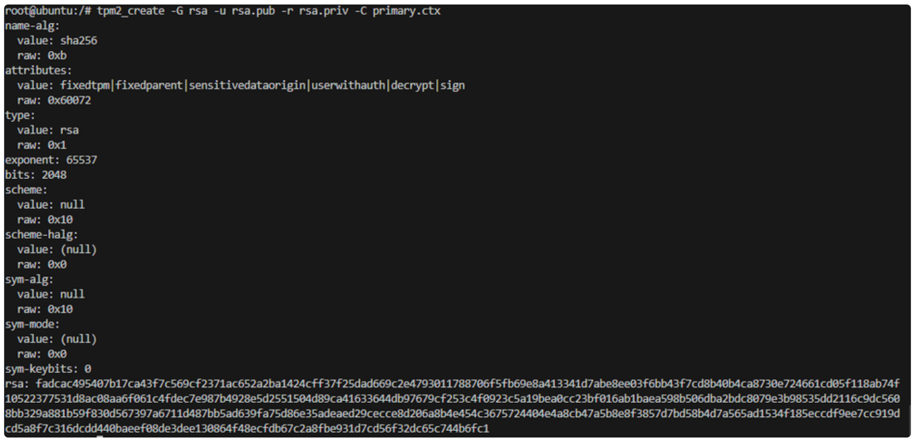
```
root@cto-k8s-wrkr-02:/home/cisadmin# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5e8117c0bd28: Pull complete
Digest: sha256:8eab65df33a6de2844c9aefd19efe8ddb87b7df5e9185a4ab73af936225685bb
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

7. Record the sha256 hash output when pulling the image file. The file hash will be used together with the TPM key to perform a signature check.
8. Create a TPM primary key. Run “tpm2_createprimary -C e -c primary.ctx”



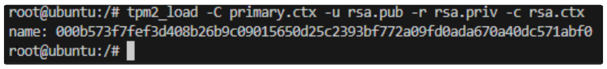
```
root@ubuntu:/# tpm2_createprimary -C e -c primary.ctx
name-alg:
  value: sha256
  raw: 0xb
attributes:
  value: fixedtpm|fixedparent|sensitivedataorigin|userwithauth|restricted|decrypt
  raw: 0x30072
type:
  value: rsa
  raw: 0x1
exponent: 65537
bits: 2048
scheme:
  value: null
  raw: 0x10
scheme-halg:
  value: (null)
  raw: 0x0
sym-alg:
  value: aes
  raw: 0x6
sym-mode:
  value: cfb
  raw: 0x43
sym-keybits: 128
rsa: d68216b7880999ac3cc0ce49c1160a8b8b0bb3b96036461bb5a0642d789dc1c99ffa909bd7f6c82c8ae9eb2df3d3d111ff1f0322ecdd0ed4d3e803ebe2e64fac9563dca770dc0
5ed0c6b7c22356f8671757205d99a476fc72c445e2dc1172bb176723dc80a355d6f3495201e4379a54240e9a4803923c73c3f1f26ba8411e46b12b98db00cd0211033da4ed3119816c
1df9e170fb5e5280d0682f4ce896316b2fcb03f801f0cd73c7bc339ce0c29583cf3d6ec12a356889b10dddf3e566670bfc36e04f778946ee210ecb6ba933481ca5ee6cedf9f10f2816
cb3d3dcb754afca4e7708b74be664ecd86b9ecdff8879f765a050493bb86fced6aa1c3694393ead
```

9. Create a child key pair as TPM requires 2 keys for hierarchical signing. Run “tpm2_create -G rsa -u rsa.pub -r rsa.priv -C primary.ctx”.



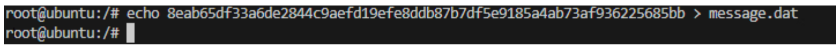
```
root@ubuntu:/# tpm2_create -G rsa -u rsa.pub -r rsa.priv -C primary.ctx
name-alg:
  value: sha256
  raw: 0xb
attributes:
  value: fixedtpm|fixedparent|sensitivedataorigin|userwithauth|decrypt|sign
  raw: 0x60072
type:
  value: rsa
  raw: 0x1
exponent: 65537
bits: 2048
scheme:
  value: null
  raw: 0x10
scheme-halg:
  value: (null)
  raw: 0x0
sym-alg:
  value: null
  raw: 0x10
sym-mode:
  value: (null)
  raw: 0x0
sym-keybits: 0
rsa: fadcac495407b17ca43f7c569cf2371ac652a2ba1424cff37f25dad669c2e4793011788706f5fb69e8a413341d7abe8ee03f6bb43f7cd8b40b4ca8730e724661cd05f118ab74f
10522377531d8ac08aa6f061c4fdec7e987b4928e5d2551504d89ca41633644db97679cf253c4f0923c5a19bea0cc23bf016ab1baea598b506dba2bdc8079e3b98535dd2116c9dc560
8bb329a881b59f830d567397a6711d487bb5ad639fa75d86e35adeaed29cecce8d206a8b4e454c3675724404e4a8cb47a5b8e8f3857d7bd58b4d7a565ad1534f185eccdf9ee7cc919d
cd5a8f7c316dcdd440baeef08de3dee130864f48ecfdb67c2a8fbe931d7cd56f32dc65c744b6fc1
```

10. Load the keys created to the TPM. Run “tpm2_load -C primary.ctx -u rsa.pub -r rsa.priv -c rsa.ctx”.



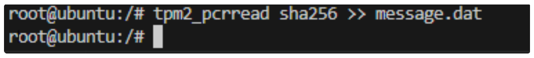
```
root@ubuntu:/# tpm2_load -C primary.ctx -u rsa.pub -r rsa.priv -c rsa.ctx
name: 000b573f7fef3d408b26b9c09015650d25c2393bf772a09fd0ada670a40dc571abf0
root@ubuntu:/#
```

11. Take the noted image hash from #8 and add it to a “message.dat file”. This is the signature to be checked. Do this by running “echo <insert_file_hash> > message.dat”.



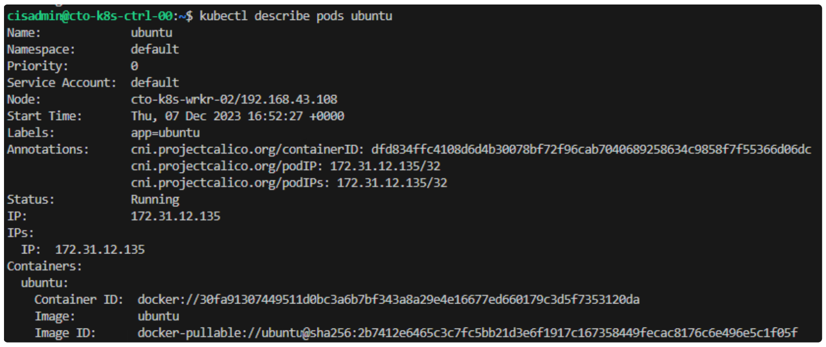
```
root@ubuntu:/# echo 8eab65df33a6de2844c9aefd19efe8ddb87b7df5e9185a4ab73af936225685bb > message.dat
root@ubuntu:/#
```

12. Run the following which bounds all the PCR values together with the image hash, “tpm2_pcrread sha256 >> message.dat”.

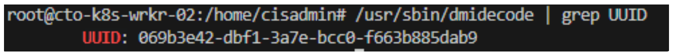


```
root@ubuntu:/# tpm2_pcrread sha256 >> message.dat
root@ubuntu:/#
```

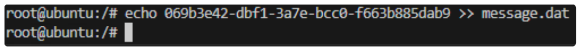
13. The VM UUID is “based on the physical computer’s identifier and the path to the virtual machine’s configuration file. The UUID is generated when you power on or reset the virtual machine. As long as you do not move or copy the virtual machine to another location, the UUID remains constant” (harrymc, 2017). Run “/usr/sbin/dmidecode | grep UUID” to obtain the UUID. Then copy the value from the node and run “ echo <UUID> >> message.dat” to append the UUID to the message.dat file to be signed. If the UUID were to change, the signature would not be verified. This command should be ran in the worker node housing the “ubuntu” container since the VM and container share a vTPM and this bounds the UUID to the appropriate worker node VM.



```
cisadmin@cto-k8s-ctrl-00:~$ kubectl describe pods ubuntu
Name:             ubuntu
Namespace:        default
Priority:         0
Service Account:  default
Node:             cto-k8s-wrkr-02/192.168.43.108
Start Time:       Thu, 07 Dec 2023 16:52:27 +0000
Labels:           app=ubuntu
Annotations:      cni.projectcalico.org/containerID: dfd834ffc4108d6d4b30078bf72f96cab7040689258634c9858f7f55366d06dc
                  cni.projectcalico.org/podIP: 172.31.12.135/32
                  cni.projectcalico.org/podIPs: 172.31.12.135/32
Status:           Running
IP:               172.31.12.135
IPs:
  IP:  172.31.12.135
Containers:
  ubuntu:
    Container ID:  docker://30fa91307449511d0bc3a6b7bf343a8a29e4e16677ed660179c3d5f7353120da
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
```

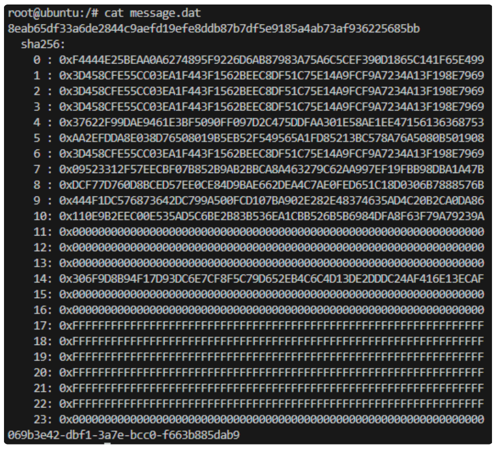


```
root@cto-k8s-wrkr-02:/home/cisadmin# /usr/sbin/dmidecode | grep UUID
        UUID: 069b3e42-dbf1-3a7e-bcc0-f663b885dab9
```



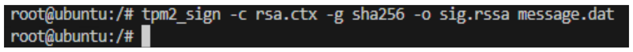
```
root@ubuntu:/# echo 069b3e42-dbf1-3a7e-bcc0-f663b885dab9 >> message.dat
root@ubuntu:/#
```

14. “cat” the message.dat file to ensure the PCR values, image hash, and UUID are present. If so, move to the next step.



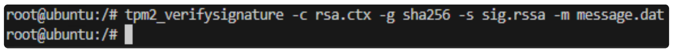
```
root@ubuntu:/# cat message.dat
8eab65df33a6de2844c9aefd19efe8ddb87b7df5e9185a4ab73af936225685bb
  sha256:
    0 : 0xF4444E25BEAA0A6274895F9226D6AB87983A75A6C5CEF390D1865C141F65E499
    1 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    2 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    3 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    4 : 0x37622F99DAE9461E3BF5090FF097D2C475DDFAA301E58AE1EE47156136368753
    5 : 0xAA2EFDDA8E038D76508019B5EB52F549565A1FD85213BC578A76A5080B501908
    6 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    7 : 0x09523312F57EECBF07B852B9AB2BBCA8A463279C62AA997EF19FBB98DBA1A47B
    8 : 0xDCF77D760D8BCED57EE0CE84D9BAE662DEA4C7AE0FED651C18D0306B7888576B
    9 : 0x444F1DC576873642DC799A500FCD107BA902E282E48374635AD4C20B2CA0DA86
    10: 0x110E9B2EEC00E535AD5C6BE2B83B536EA1CBB526B5B6984DFA8F63F79A79239A
    11: 0x0000000000000000000000000000000000000000000000000000000000000000
    12: 0x0000000000000000000000000000000000000000000000000000000000000000
    13: 0x0000000000000000000000000000000000000000000000000000000000000000
    14: 0x306F9D8B94F17D93DC6E7CF8F5C79D652EB4C6C4D13DE2DDDC24AF416E13ECAF
    15: 0x0000000000000000000000000000000000000000000000000000000000000000
    16: 0x0000000000000000000000000000000000000000000000000000000000000000
    17: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    18: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    19: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    20: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    21: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    22: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    23: 0x0000000000000000000000000000000000000000000000000000000000000000
069b3e42-dbf1-3a7e-bcc0-f663b885dab9
```

14. Sign the message by running "tpm2_sign -c rsa.ctx -g sha256 -o sig.rssa message.dat".



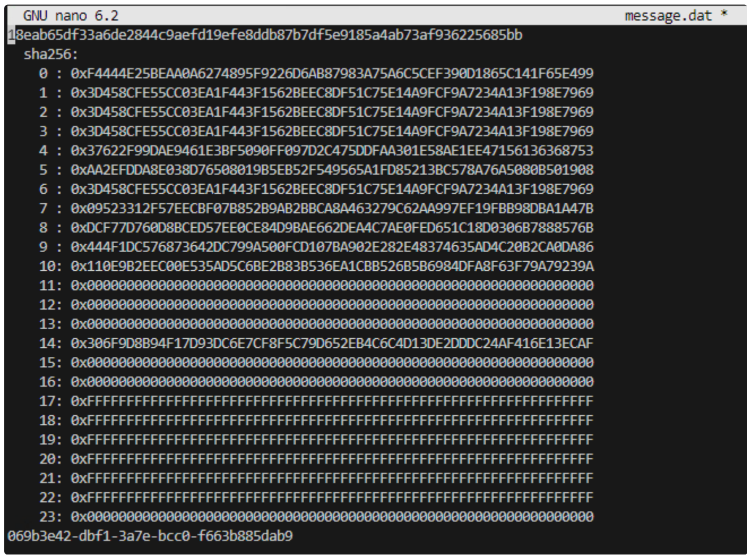
```
root@ubuntu:/# tpm2_sign -c rsa.ctx -g sha256 -o sig.rssa message.dat
root@ubuntu:/#
```

15. Verify the signature as unchanged. Run "tpm2_verifysignature -c rsa.ctx -g sha256 -s sig.rssa -m message.dat". Please note that if the image hash has NOT changed, there will be NO return.



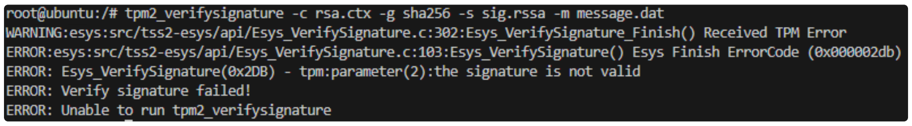
```
root@ubuntu:/# tpm2_verifysignature -c rsa.ctx -g sha256 -s sig.rssa -m message.dat
root@ubuntu:/#
```

16. To test whether the signature will be verified if any message.dat values changed, start by editing the "message.dat" file. Add an arbitrary character such as "1" that was added in the screenshot.



```
  GNU nano 6.2                                                message.dat *
18eab65df33a6de2844c9aefd19efe8ddb87b7df5e9185a4ab73af936225685bb
  sha256:
    0 : 0xF4444E25BEAA0A6274895F9226D6AB87983A75A6C5CEF390D1865C141F65E499
    1 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    2 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    3 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    4 : 0x37622F99DAE9461E3BF5090FF097D2C475DDFAA301E58AE1EE47156136368753
    5 : 0xAA2EFDDA8E038D76508019B5EB52F549565A1FD85213BC578A76A5080B501908
    6 : 0x3D458CFE55CC03EA1F443F1562BEEC8DF51C75E14A9FCF9A7234A13F198E7969
    7 : 0x09523312F57EECBF07B852B9AB2BBCA8A463279C62AA997EF19FBB98DBA1A47B
    8 : 0xDCF77D760D8BCED57EE0CE84D9BAE662DEA4C7AE0FED651C18D0306B7888576B
    9 : 0x444F1DC576873642DC799A500FCD107BA902E282E48374635AD4C20B2CA0DA86
    10: 0x110E9B2EEC00E535AD5C6BE2B83B536EA1CBB526B5B6984DFA8F63F79A79239A
    11: 0x0000000000000000000000000000000000000000000000000000000000000000
    12: 0x0000000000000000000000000000000000000000000000000000000000000000
    13: 0x0000000000000000000000000000000000000000000000000000000000000000
    14: 0x306F9D8B94F17D93DC6E7CF8F5C79D652EB4C6C4D13DE2DDDC24AF416E13ECAF
    15: 0x0000000000000000000000000000000000000000000000000000000000000000
    16: 0x0000000000000000000000000000000000000000000000000000000000000000
    17: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    18: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    19: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    20: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    21: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    22: 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    23: 0x0000000000000000000000000000000000000000000000000000000000000000
069b3e42-dbf1-3a7e-bcc0-f663b885dab9
```

17. Run "tpm2_verifysignature -c rsa.ctx -g sha256 -s sig.rssa -m message.dat" again to verify the signature.



```
root@ubuntu:/# tpm2_verifysignature -c rsa.ctx -g sha256 -s sig.rssa -m message.dat
WARNING:esys:src/tss2-esys/api/Esys_VerifySignature.c:302:Esys_VerifySignature_Finish() Received TPM Error
ERROR:esys:src/tss2-esys/api/Esys_VerifySignature.c:103:Esys_VerifySignature() Esys Finish ErrorCode (0x000002db)
ERROR: Esys_VerifySignature(0x2DB) - tpm:parameter(2):the signature is not valid
ERROR: Verify signature failed!
ERROR: Unable to run tpm2_verifysignature
```

As expected, the signature could not be verified by the TPM because the image hash was changed. Not only will changes to the file hash result in no verification, PCR changes will also return the same result. Using IMA, policies can be created to measure when OS changes occur on PCRs 10 and 12. The policies, if crafted appropriately, could cover a large part of variable benchmark recommendations. The TPM provides assurance to the verification of signatures by tying in PCR changes not logged by the signing key.

References

General Commands Manual. tpm2_verifysignature(1) - Arch manual pages. (n.d.). https://man.archlinux.org/man/tpm2_verifysignature.1.en

harrymc. (2017, May 15). Does a computer (even a virtual one) have a property which identifies it? https://superuser.com/questions/1209273/does-a-computer-even-a-virtual-one-have-a-property-which-identifies-it

TCG. (2019, November). TPM 2.0 Library. Trusted Computing Group. https://trustedcomputinggroup.org/resource/tpm-library-specification/