HOST INTEGRITY
AT RUNTIME
& STARTUP

Host Integrity At Runtime & Startup (HIRS)

TCG Rim Tool

October 2024

Users Guide

Version 3.0

Table of Contents

This page left intentionally blank.

Purpose

The purpose of this document is to support and define a command line application called the tcg_rim_tool. The tcg_rim_tool can be used to create SWID tags that adhere to the TCG PC Client RIM specification. It supports the ability to digitally sign and verify the signature of a Base RIM. The Base RIM may be uploaded to the HIRS ACA if a valid signature is evident.

Background

UEFI Boot

During the boot process of a UEFI compatible device, Firmware records cryptographic hashes to the Trusted Platform Module (TPM). These hashes normally capture firmware modules, firmware configuration, expansion component firmware, expansion component firmware configurations, and the bootloader. TPM-aware bootloaders can continue logging hashes to describe the kernel, initial file system, and any modules. Kernels, applications, and drivers can also log runtime hashes to the TPM.

Hashes are stored in the TPM's Platform Configuration Registers (PCRs) in accordance with Figure 1. Most TPMs have 24 PCRs per supported hash algorithm. TPM 2.0 supports SHA-1 and SHA-256 at the minimum (48 PCRs minimum). PCR values are computed via a series of one-way hashes where each measurement hash is appended to the current PCR value, and then the combination is hashed and becomes the new PCR value (referred to as "extending the PCR").

UEFI also records measurement hashes, along with information about these hashes, in the TPM Event Log. The TPM Event Log is an audit log that can be used for verification later, after the system boots.

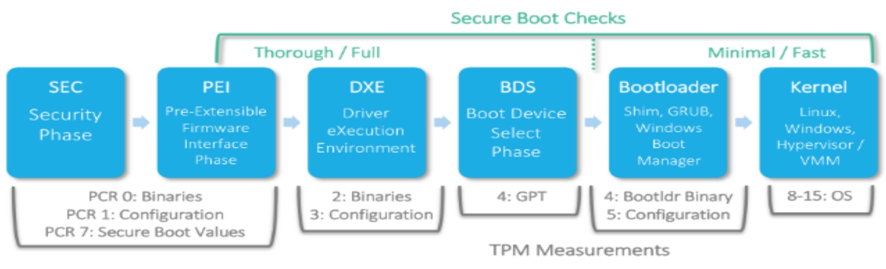

Figure 1: The interaction of Secure Boot and TPM with UEFI boot phases

TPM Event Log

The TPM Event Log is defined in the TCG PC Client Platform Firmware Profile which is referred to as the "PFP". The Event Log file contains all the hashes that get extended into the TPM Platform Configuration Registers (PCR) during the boot cycle, as well as details about each hash and each hash's corresponding event. One can recreate the resultant PCR values by extending the values within this file, in which case the TPM PCR list may not be needed.

The project Host Integrity at Runtime and Startup (HIRS) contains a server-side application referred to as the Attestation Certificate Authority (ACA). The ACA uses the Event Log during its validation process if the firmware option is selected. The Event Log is one of the Support RIM file options for PC Client systems. This means that the Base RIM (SWID tag) file will have a hash of the Event Log in its payload for verification purposes. During the HIRS validation process, the Event Log is used to prove two things:

Real-time values check: The digest values found within the Event Log are used to calculate a composite hash, which the ACA compares against the expected composite hash value in the TPM Quote. Should TPM Quote verification pass, this proves that the ACA has received the correct Event Log and nothing has been altered in the Event Log since it was created (we already know the Quote is good based on the signature and nonce). Should TPM Quote verification fail, the Event Log file is needed to provide details on the individual hashes within each PCR.

Reference values check: Each event in the Event Log (supplied by the client) is compared against the events listed in the RIM (supplied by the OEM) to provide details in the case of events failing comparison.

Reference Integrity Manifests

The TCG Reference Integrity Information Model (RIM IM) defines structures that a Verifier (i.e. a system that analyzes evidence from a platform or platform component to determine it's state) uses to validate expected values (Assertions) against actual values (Evidence).

The PC Client RIM specification is considered a RIM IM "binding specification" specifically geared for PC Clients and Servers. An OEM produced PC Client RIM can be used by the HIRS ACA when the Firmware Validation Policy option is enabled. Firmware Validation compliments the Platform Certificate for Supply Chain acceptance testing by providing an automated means to verify the firmware and boot software for the platform before an Attestation Certificate will be issued.

RIM Bundles

For the PC Client, there are two different types of RIM files: the Base RIM and the Support RIMs. This is designated by the TCG as the "RIM Bundle".

The Base RIM

Base RIM: The PC Client RIM defines the Base Rim as a ISO 197770-2 Software Identity (SWID) standard compatible file. The Base RIM provides a verifiable identity of the RIM creator and also integrity information of Support RIMs. The Base RIM contains:

- Cryptographically verifiable identification of the Creator of the RIM and Support RIMs.
- A unique identifier (tagId) for a set of RIM Bundles.
- A reference to the binding specification that defines the Support RIMs.
- Cryptographic hashes (digests) of all Payload references including Support RIMs.

- A digital signature of the RIM signed by the RIM's Creator.

The Support RIM

Support RIM - The TPM Event Log: A support RIM contains assertions about the state or configuration of the platform to which the RIM applies (a.k.a., Reference Integrity Measurements).

For PC Clients, the Support RIM utilizes the **TPM Event Log** created during the boot process. The TPM log defined by the TCG PC Client Platform Firmware Profile and captures all events that extend any of the TPMs Platform Configuration Register (PCR) contents. The OEM that creates the RIM captures the event log at the end of the production process and inserts a hash of the log into the Base RIM before the Base RIM is signed. It then stores the RIM onto the device or optionally provides a Uniform Resource Identifier (URI).

Linux has support for reading the log and writing it to the securityfs partition for TPM. You will need a Linux kernel of 4.18 or higher to see the log in the following path:
/sys/kernel/security/tpm0/binary_bios_measurements

Windows 10 MAY(depending upon specific version and group policy) store of the binary form of the event log in C:\Windows\Logs\MeasuredBoot\ with a numeric name and a .log extension.

RIM Lifecycle

The SWID specification defines four types of SWID tags: corpus, primary, patch, and supplemental. All four tag types come into play at various points in the software lifecycle, and support software management processes that depend on the ability to accurately determine where each software product is in its lifecycle. Corpus Tags as associated with a software installation package. RIM support for corpus is currently undefined and not supported by this tool.

Primary Base RIM

Primary Base RIM should include all necessary reference measurements needed to verify the Firmware (PCR 0 -7). The Primary Base RIM is expected to be part of an initial RIM Bundle.

Patch Base RIM

Patch Base RIM is intended to accompany Firmware updates and include any reference measurements necessary to cover the changes provided by the firmware update.

Supplemental Base RIM

Supplemental Base RIM: Supplemental RIM are intended to support RIMs created by organizations other than the OEM (e.g. System Integrator's or Value-Added Reseller's (VARs)). The signature on the base RIM is expected to signed by the organization that created the supplemental Base RIM. The

Supplemental uses the Href element to point to the Primary or Patch RIM that the supplemental RIM is adding Reference Measurements for.

Composite RIMs

Composite RIM: A RIM Bundle that includes or references other Base RIM Instances in its payload element

There may be scenarios in which multiple entities take part in the production of a given device. That in turn may lead the Verifier to retrieve multiple RIM Bundles in order to verify the device. Such a scenario may require a RIM Bundle associated with the device to include or provide references to other RIM Bundle(s) being managed by other entities.

A Base RIM that includes or references other Base RIMs is a Composite RIM. Consider a modern PC manufacturer that includes components from various component vendors (e.g., disk drive,memory, CPU's, etc.). Each component vendor may have its own RIM that corresponds to Firmware running on the component. The PC manufacturer may wish to include or reference a component RIM in its own RIM without corrupting the original component RIM's signature. The PC Manufacturer may also want its own signature on the RIM to include coverage of all the component RIMs. The inclusion of Component RIM reference within a PC manufacturers RIM is illustrated in the following:

```
PC_BaseRIM
        |-------> PC_Support RIM
        |-------> Component1_BaseRIM
        |         |-------->Component1_Support RIM
        |-------> Component2_BaseRIM
        |         |-------->Component2_Support RIM
        |               |-------->SubComponentA_BaseRIM
        |                           |-------->SubComponentA_Support RIM End
    Done
```

Note that the tcg_rim_tool process 1 Base RIM at a time. Each PC Base RIM or Component RIM must be created or verified at time.

Timestamps

A RIM signer may include a s PC Client RIM optionally supports a simple timestamp or a countersignature timestamp. Time stamp services are typically provided by a commercial Time Stamp Authority (TSA). Most TSAs support the RFC 3161 Time-Stamp Protocol but the time stamp itself may be encoded using a number of methods.

RFC3339 Timestamps

The tgc_rim_tool will support the creation of a RFC3339 based timestamp. The format of the time is: yyyy-MM-ddThh:mm:ssZ

RFC3852 Timestamps

The RFC3852 includes support for a countersignature. The tgc_rim_tool does not directly support the creation of a countersignature but will accept a countersignature file created by a third party application an insert it into a timestamp field within the Base RIM.

Layered Endorsements (detached signatures)

In some cases, there may be a need for an entity that is not the tag creator to provide a secondary signature for the RIM (e.g., a layered endorsement). The PC Client RIM utilizes the W3C defined "Detached Signature" for providing secondary signatures to be applied to an existing signed PC Client RIM. The initial Base RIM uses the enveloped signature provided by the tagCreator. Each secondary signature will be a sibling element of the BaseRIMs SoftwareIdentity element. The secondary signature uses the Base RIM's tagId as the SignedInfo Reference.

The tcg_rim_tool does not directly support the creation of a secondary signature but will accept a secondary signature file created by a third party application an insert it into a timestamp field within the Base RIM.

Installing the tcg_trim_tool

Please refer tot he tcg_rim_tool reame for intructions on how to install the latest version of the tcg_rim_tool: https://github.com/nsacyber/HIRS/tree/main/tools/tcg_rim_tool

Using the tcg_rim_tool

Files

The tcg_rim_tool is a command line tool that works off of a set of files:

attributes: The json formatted configuration file holding attributes used to populate the base RIM. Refer to appendix A for an example of the rim_fields.json file.

privateKeyFile: The PEM formatted private key file used to sign the base RIM created by the create function.

publicCertificate: The PEM formatted public key certificate used to verify a RIM file or to embed in a signed RIM.

truststore: The PEM formatted truststore to sign the base RIM created or to validate the signed base RIM.

rimel: The binary TPM Event Log file to use as a support RIM.

Commands

Create

The create command is used to format and sign a Base RIM. The create command allows for a verification certificate to be embedded in a swidtag or not include an embedded certificate (the default)

To create a Base RIM you need:

- An attributes file which holds the attributes to populate the Base RIM
- A rimel file which holds a TPM Event Log file to use as a Support RIM
- A privateKeyFile file containing the private key used to sign the Base RIM
- A publicCertificate file containing the public key certificate to be embedded into the Base RIM (optional)url

The command for creating rim without a embedded signature certificate is:

```
rim –c base –a rim_fields.json -l TpmLog.bin -k privateRimKey.pem -o base_rim.swidtag
```

The tcg_rim_tool will create the Base RIM and it will be written out to the base_rim.swidtag file.

The command for creating rim with an embedded signature certificate is:

```
rim –c base –a rim_fields.json -l TpmLog.bin -k privateRimKey.pem -o base_rim.swidtag -p
publicCertificate.pem
```

Notes

- The json file will need to be properly formatted to avoid unnecessary errors when the tool parses the file. Its a good idea to use an online json lint or verification tool to check that the json file is properly formatted.
- All values within the json file are used as input for the Base RIM creation.
- If the size is empty the tool will automatically populate the size attribute.
- If the hash attribute is empty the tool will automatically populate the hash. he Base RIM is a signed SWID tag instance that provides a verifiable identity of the RIM creator plus integrity information of one or more support RIM(s). The Base RIM contains a digest of each support RIM. The Base RIM also contains a signature.

Primary RIM Creation

For Swidtags a Primary tag is assumed when Patch and Supplemental attributes are set to false. To set this in the attributes file ensure that the "SoftwareIdentity" element has the "patch" and "supplemental" attributes set to false:

```
"SoftwareIdentity": {
            "name": "Example.com BIOS",
            "version": "01",
            "tagId": "94f6b457-9ac9-4d35-9b3f-78804173b65as",
            "tagVersion": "0",
            "patch": false,
            "supplemental": false
      }
```

Patch RIM creation

To create a Patch Base RIM set the "SoftwareIdentity" element "patch" attribute to true in the attributes file.

A Patch RIM should to have a link back to the tagId of the Previous RIM version. To accomplish this take the tagId (should be a guid) and place this in the href value

```
"Link": {
            "href": "94f6b457-9ac9-4d35-9b3f-78804173b65as",
            "rel": ”patches”
      },
```

A Patch RIM is required to have the RIMLinkHash attribute present. The RIMLinkHash specifies the digest of the previous Base RIM. In many cases that will be the Primary Base RIM.

The RIMLinkHash is a base64 encoded SHA256 digest of the RIM referenced by the Link element (Href rel=”supersedes” ,rel=”patches, or rel=”requires”). Note that the Primary Base RIM does not require RIMLinkHash.

To create a RIMLinkHash take a sha256 hash of the previous RIM and insert it into the RIMLinkHash in te Meta element in the in the attributes file:

```
"Meta": {
     ...,
     "RIMLinkHash": "4479ca722623f8c47b703996ced3cbd981b06b1ae8a897db70137e0b7c546848"
    }
```

Supplemental RIM Creation

To create a Supplemental Base RIM set the "SoftwareIdentity" element "supplemental" attribute to true in the attributes file.

A Supplemental RIM should to have a link back to the tagId of the Previous RIM version. To accomplish this take the tagId (should be a guid) and place this in the href value in similar manner as the Patch RIM.

A Supplemental RIM is required to have the RIMLinkHash attribute present. The Meta element in the in the attributes file should have the RIMLinkHash sha256 hash of the previous RIM filled out similar to the Patch RIM. The Base RIM is a signed SWID tag instance that provides a verifiable identity of the RIM creator plus integrity information of one or more support RIM(s). The Base RIM contains a digest of each support RIM. The Base RIM also contains a signature.

Composite RIM Creation

A Composite Base RIM has one or more payloads which refer to other base RIMs. To create a composite rim edit the attributes file payload element and enter a new payload wit the required supportRimFormat, supportRimType, or (optional) supportRimUriGlobal:

```
"Payload": {
            "supportRIMURIGlobal": "https://Example.com/support/ProductA/firmware/rims/",
            "supportRIMFormat":"TCG_EventLog_Assertion",
            "supportRimType": "BaseRim",
            "Directory": {
                  "name": "rim",
                  "root": "/boot/efi/EFI/tcg/manifest/rim/",
                  "File": {
                        "version":"01",
                        "name": "Example.com.BIOS.01.rimel",
                        "size": "7549",
                        "hash":
"4479ca722623f8c47b703996ced3cbd981b06b1ae8a897db70137e0b7c546848"
                        }
            }
      }
```

Adding an RFC3339 timestamp

To add an RFC3339 timestamp to the Base RIM use the optional --timestamp parameter with the time forammted as "yyyy-MM-ddThh:mm:ssZ"

```
rim –c base –a rim_fields.json -l TpmLog.bin -k privateRimKey.pem -o base_rim.swidtag --timestamp RFC3339 "20204-09-01T10:00:00Z"
```

Adding an RFC3852 timestamp

Adding an RFC 3852 based timestamp requires a third party tool capable of producing an RFC 3852 based timestamp as a file. Once the RFC 3852 based timestamp file is created it can be optionally added to the creation of the Base RIM

```
rim –c base –a rim_fields.json -l TpmLog.bin -k privateRimKey.pem -o base_rim.swidtag --timestamp RFC3852 timestamp.bin"
```

Verify

The Verify command is used to check the signature on the Base RIM as well as the payload hashes (hash taken of the support rim file(s).

To verify a RIM you need:

- A rimel file which holds a TCG Event Log file to use as a Support RIM
- A truststore file containing the certificate chain key used to verify the publicCertificate which is in turn used to verify the signature of the Base RIM
- A publicCertificate file containing the public key certificate if the Base RIM does not include an embedded certificate (optional)

The command for verifying the Base RIM is

```
rim -v base_rim.swidtag -l TpmLog.bin -t truststore.pem
```

If the Base RIM does not include an embedded cert and the cert is not part of the truststore file it may be necessary to include the certificate: The command for verifying the Base RIM is

```
rim -v base_rim.swidtag -l TpmLog.bin -t truststore.pem -p publicCertificate
```

Appendix A Example attribute json file for a Primary RIM

The following is an example of the contents of the attributes json file used by the tcg_rim_tool. It is used to create a Primary Base RIM.

```
{
    "SoftwareIdentity": {
        "name": "Example.com BIOS",
        "version": "01",
        "tagId": "94f6b457-9ac9-4d35-9b3f-78804173b65as",
        "tagVersion": "0",
        "patch": false,
        "supplemental": false
    },
       "Entity": {
        "name": "Example Inc",
        "regid": "http://Example.com",
        "role": "softwareCreator,tagCreator"
    },
       "Link": {
        "href": "https://Example.com/support/ProductA/firmware/installfiles",
        "rel": "installationmedia"
    },
       "Meta": {
        "colloquialVersion": "Firmware_2019",
        "edition": "12",
        "product": "ProductA",
        "revision": "r2",
        "PayloadType": "direct",
        "platformManufacturerStr": "Example.com",
        "platformManufacturerId": "00201234",
        "platformModel": "ProductA",
        "platformVersion": "01",
        "firmwareManufacturerStr": "BIOSVendorA",
        "firmwareManufacturerId": "00213022",
        "firmwareModel": "A0",
        "firmwareVersion": "12",
        "bindingSpec": "PC Client RIM",
        "bindingSpecVersion": "1.2",
        "pcURIlocal": "/boot/tcg/manifest/switag/",
        "pcURIGlobal": "https://Example.com/support/ProductA/"
    },
       "Payload": {
        "supportRIMURIGlobal": "https://Example.com/support/ProductA/firmware/rims/",
            "supportRIMFormat":"TCG_EventLog_Assertion",
            "supportRimType": "Indirect",
            "Directory": {
                  "name": "rim",
                  "root": "/boot/tcg/manifest/rim/",
                  "File": {
                        "version":"01",
                        "name": "Example.com.BIOS.01.rimel",
                        "size": "7549",
                        "hash":
"4479ca722623f8c47b703996ced3cbd981b06b1ae8a897db70137e0b7c546848"
                        }
            }
      }
}
```

Appendix B: Example Primary base RIM (XML)

The following is an example output of the tcg_rim_tool. It is a signed Primary Base RIM example that contains an embedded certificate.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SoftwareIdentity xmlns="http://standards.iso.org/iso/19770/-2/2015/schema.xsd"
xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
 name="Example.com BIOS"
 corpus="false"
 patch="false"
 supplemental="false"
 tagId="94f6b457-9ac9-4d35-9b3f-78804173b65as"
 tagVersion="0"
 version="01"
 versionScheme="multipartnumeric" xml:lang="en">
 <Entity
    name="Example Inc"
    regid="http://Example.com"
    role="softwareCreator tagCreator"/>
 <Link href="https://Example.com/support/ProductA/firmware/installfiles" rel="installationmedia"/>
 <Meta xmlns:n8060="http://csrc.nist.gov/ns/swid/2015-extensions/1.0"
      xmlns:rim="https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model"
      n8060:colloquialVersion="Firmware_2019"
      n8060:edition="IOT"
      n8060:product="ProductA"
      n8060:revision="r2"
   rim:bindingSpec="PC Client RIM"
   rim:bindingSpecVersion="1.2"
   rim:pcURIGlobal="https://Example.com/support/ProductA/firmware/rims"
   rim:platformManufacturerId="00213022"
   rim:platformManufacturerStr="BIOSVendorA" rim:platformModel="A0" rim:platformVersion="12"/>
 <Payload>
   <Directory name="rim">
       <File xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
           SHA256:hash="4479ca722623f8c47b703996ced3cbd981b06b1ae8a897db70137e0b7c546848"
           name="Example.com.iotBase.bin" size="7549"/>
   </Directory>
 </Payload>
 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
  <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
  <Reference URI="">
```

```
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <DigestValue>fze9UG1Ft9l80Yn8Z4oDTy7G0iCtk+y7hIfoOru6pDI=</DigestValue>
   </Reference>
 </SignedInfo>
<SignatureValue>b21c8nvkjYO0MZvm8quOlLkd/ocTrdpQ55G7mKELy4wDHRxKe3LLdKXOFpab9A9oTwtd
EXZnHTCahwdt31YALMuTSY4kBORuPDsaby/Cte/35/2gGwkZEEE1DNBVXAG97SiuBd5koebcT8TTdGGis6w
rUzcDnFXEt3LLAU8ZHhl7weqeyuqsNP12teCpb2Ru9FDWBOUjgOeBo7P6qdJJCG3txmsD1pjA92zg
zuLzwiY1B+sPk8aC5n9LiXOgaDr2MLuijmIrGJzOYEItgMQ+a5ncyZtb8hHdC93xnk0IlnaG5wuv1Ribeg/Zr6z5
k4Yg6Z+ErOPqlDSyPJMZEJZdkQ==
</SignatureValue>
 <KeyInfo>
    <KeyName>2fdeb8e7d030a2209daa01861a964fedecf2bcc1</KeyName>
    <KeyValue>
      <RSAKeyValue>

<Modulus>p3WVYaRJG7EABjbAdqDYZXFSTV1nHY9Ol9A5+W8t5xwBXBryZCGWxERGr5AryKWPxd+qzjj+cF
pxxkM6N18jEhQIx/CEZePEJqpluBO5w2wTEOe7hqtMatqgDDMeDRxUuIpP8LGP00vh1wyDFFew90d9
dvT3bcLvFh3a3ap9bTm6aBqPup5CXpzrwIU2wZfgkDytYVBm+8bHkMaUrgpNyM+5BAg2zl/Fqw0q
otjaGr7PzbH+urCvaGbKLMPoWkVLIgAE8Qw98HTfoYSFHC7VYQySrzIinaOBFSgViR72kHemH2lW
 jDQeHiY0VIoPik/jVVIpjWe6zzeZ2S66Q/LmjQ==
        </Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
     </KeyValue>
 <X509Data>

<X509SubjectName>CN=example.RIM.signer,OU=PCClient,O=Example,ST=VA,C=US</X509SubjectName
>
      <X509Certificate>MIIDoTCCAomgAwIBAgIJAPB+r6VBhBn5MA0GCSqGSIb3DQEBCwUAMFMxCzA
JBgNVBAYTAlVTMQswCQYDVQQIDAJWQTEQMA4GA1UECgwHRXhhbXBsZTERMA8GA1UECwwIUENDbGl
lbnQxEjAQBgNVBAMMCUV4YW1wbGVDQTAeFw0yMDAzMTExODExMjJaFw0zMDAxMTgxODExMjJaMF
wxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJWQTEQMA4GA1UECgwHRXhhbXBsZTERMA8GA1UECwwIUE
NDbGllbnQxGzAZBgNVBAMMEmV4YW1wbGUuUklNLnNpZ25lcjCCASIwDQYJKoZIhvcNAQEBBQADggE
PADCCAQoCggEBAKd1lWGkSRuxAAY2wHag2GVxUk1dZx2PTpfQOflvLeccAVwa8mQhlsRERq+QK8i
lj8Xfqs44/nBaccZDOjdfIxIUCMfwhGXjxCaqZbgTucNsExDnu4arTGraoAwzHg0cVLiKT/Cxj9N
L4dcMgxRXsPdHfXb0923C7xYd2t2qfW05umgaj7qeQl6c68CFNsGX4JA8rWFQZvvGx5DGlK4KTcj
PuQQINs5fxasNKqLY2hq+z82x/rqwr2hmyizD6FpFSyIABPEMPfB036GEhRwu1WEMkq8yIp2jgRU
oFYke9pB3ph9pVow0Hh4mNFSKD4pP41VSKY1nus83mdkuukPy5o0CAwEAAaNvMG0wHQYDVR0OBBY
EFC/euOfQMKIgnaoBhhqWT+3s8rzBMB8GA1UdIwQYMBaAFEahuO3bpnFf0NLneoo8XW6aw5Y4MAk

GA1UdEwQCMAAwCwYDVR0PBAQDAgbAMBMGA1UdJQQMMAoGCCsGAQUFBwMDMA0GCSqGSIb3DQ
```

```
EBCwUAA4IBAQBl2Bu9xpnHCCeeebjx+ILQXJXBd6q5+NQlV3zzBrf0bleZRtsOmsuFvWQoKQxsfZuk7Qc
SvVd/1v8mqwJ0PwbFKQmrhIPWP+iowiBNqpG5PH9YxhpHQ1osOfibNLOXMhudIQRY0yAgqQf+MOl
XYa0stX8gkgftVBDRutuMKyOTf4a6d8TUcbG2RnyzO/6S9bq4cPDYLqWRBM+aGN8e00UWTKpBl6/
1EU8wkJA6WdllK2e8mVkXUPWYyHTZ0qQnrYiuLr36ycAznABDzEAoj4tMZbjIAfuscty6Ggzxl1W
byZLI6YzyXALwaYvrcrTLeyFynlKxuCfDnr1SAHDM65BY
      </X509Certificate>
   </X509Data>
 </KeyInfo>
</Signature>
</SoftwareIdentity>
```

Appendix C: tcg_rim_tool data files

The following files are installed under /opt/rimtool/data with the installation of tcg_rim_tool. They can serve as examples of properly formed files used during rimtool operations.

rim_fields.json
The configuration file from which base RIM attributes are set. It can be passed to the `-a|--attributes` command line option.

keystore.jks
The default Java keystore referenced by the `-d|--default` command line option. It contains a single self-signed root cert with 2048-bit RSA key and is not password-protected. It is not recommended for production use.

privateRimKey.pem
A private key in PEM format. It corresponds to the public key stored in RimSignCert.pem and can be passed to the `-k|--privateKeyFile` command line option.

RimSignCert.pem
A public key in PEM format. It corresponds to the private key stored in privateRimKey.pem and can be passed to the `-p|--publicCertificate` command line option.

RimCertChain.pem
A PEM format CA chain containing the self-signed root cert in keystore.jks and the public key in RimSignCert.pem, and can be passed to the `-t|--truststore`

testCsv.swidtag
An example CSV formatted report with a Base RIM.

Examplecsv.csv
An example CSV formatted verification report.

TPMLog_Altered.bin
An event log file with an altered digest within an event that can serve as a test pattern for support RIM intended to create a failed verification.

TpmLog.bin
An event log file that can serve as the support RIM for creating a base RIM. It can be passed to the `-l|--rimel` command line option.

generated_default_cert.swidtag
A valid base RIM created from the options `-l TpmLog.bin -d`.

generated_user_cert.swidtag
A valid base RIM created from the options '-l TpmLog.bin -k privateRimKey.pem -p RimSignCert.pem'.

generated_user_cert_embed.swidtag
A valid base RIM created from the options '-l TpmLog.bin -k privateRimKey.pem -p RimSignCert.pem -e', resulting in a file similar to generated_user_cert.swidtag, with the signing cert included in the signature.

generated_timestamp_rfc3339.swidtag
A valid base RIM created from the options '-l TpmLog.bin -k privateRimKey.pem -p RimSignCert.pem', resulting in a file similar to generated_user_cert.swidtag, with a timestamp in RFC3339 format included in the signature block.

generated_timestamp_rfc3852.swidtag
A valid base RIM created from the options '-l TpmLog.bin -k privateRimKey.pem -p RimSignCert.pem', resulting in a file similar to generated_user_cert.swidtag, with a timestamp in RFC3852 format included in the signature block.

counterSignature.file
The counter signature file used to create generated_timestamp_rfc3852.swidtag.

Appendix D: References

[1] Trusted Computing Group. (2024, April 26). *TCG PC Client Reference Integrity Manifest Specification, Vs 1.1 Rev 1.0*. Retrieved from Trusted Computing Group: https://trustedcomputinggroup.org/resource/tcg-pc-client-reference-integrity-manifest-specification/

[2] Trusted Computing Group. (2024, April 26). *TCG Reference Integrity Manifest (RIM) Information Model, Vs 1.1 Rev 1.0*. Retrieved from Trusted Computing Group: https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model/

[3] Trusted Computing Group. (2023, Dec 4). *TCG PC Client Platform Firmware Profile Specification, Vs 1.06 Rev 52*. Retrieved from Trusted Computing Group: https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[4] NSACyber HIRS Group. (2024, Aug 1). *ACA User Guide*. Retrieved from Github HIRS: https://github.com/nsacyber/HIRS/tree/main?tab=readme-ov-file#quick-links

[5] Cybersecurity Requirements Center. (2019, June). *Boot Security Modes and Recommendations*. Retrieved from National Security Agency: https://media.defense.gov/2019/Jul/16/2002158058/-1/-1/0/CSI-BOOT-SECURITY-MODES-AND-RECOMMENDATIONS.PDF