TRUSTED® COMPUTING GROUP

SPECIFICATION

TCG Reference Integrity Manifest (RIM) Information Model

Version 1.1
Revision 1.0
April 26, 2024

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

Acknowledgement

The TCG wishes to thank those who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Special thanks to the members of the IWG group and others contributing to this document:

A.J. Stein	United States Government
Adonay Behre	AMI
Adrian Shaw	HP Inc.
Alberto Munoz	Intel Corporation
Amit Kapoor	Integrity Security Services, LLC
Amy Nelson	Dell, Inc.
Anas Arif	Intel Corporation
Antonio Javier Cabrera Gutierrez	Infineon Technologies
Bill Keown	Lenovo (United States) INC
Bill Sulzen	Cisco Systems
Brad Litterell	Microsoft
Brandon Weeks	Google Inc.
Carolin Baumgartner	Carolin Baumgartner
Chris Fenner	Google Inc.
Dan Morav	Nuvoton Technology Corporation
Dana Amsalem Cohen	Nuvoton Technology Corporation
Daniel Wong	Google Inc.
David Challener	David Challener
David Koplos	Micron Technology, Inc
David Safford	David Safford
Dennis Mattoon	Microsoft
Dick Wilkins	Phoenix Technologies
Donna Yu	Microsoft
Doug Ambrisko	Cisco Systems
Endrigo Pinheiro	HP Inc.
Eoin Carroll	Toyota Motor Corporation
Eric Hibbard	Samsung Semiconductor Inc.
Fabien Arrive	STMicroelectronics
Ferdinand Nolscher	Google Inc.
Fernando Tavares	Positivo Tecnologia S.A.
Florian Schweiger	Infineon Technologies
Frederick Otumfuor	AMI
Ga-Wai Chin	Infineon Technologies
Gongyuan Zhuang	Advanced Micro Devices, Inc.
Graeme Proudler	Graeme Proudler
Greg Kazmierczak	Greg Kazmierczak
Guy Fedorkow	Juniper Networks, Inc.
Henk Birkholz	Fraunhofer Institute for Secure Information Technology (SIT)
Isaac Asay	Advanced Micro Devices, Inc.
Jason Kolodziej	Dell, Inc.

Jason Young	Dell, Inc.
Jean Fioretti	WISeKey Semiconductors
Jeff Andersen	Google Inc.
Jen ye	Advanced Micro Devices, Inc.
Jerry Vacek	United States Government
Jesse Pool	Broadcom
Jiewen Yao	Intel Corporation
Joe Pennisi	NVIDIA Corporation
Jonathan Wilbur	Wildboar Software
Joshua Schiffman	HP Inc.
Ken Goldman	IBM
Lawrence Reinert	United States Government
Liqun Chen	University of Surrey
Ludovic Jacquin	NVIDIA Corporation
Michael Eckel	Fraunhofer Institute for Secure Information Technology (SIT)
Mike Boyle	United States Government
Monty Wiseman	Beyond Identity
Mukund Khatri	Dell, Inc.
Ned Smith	Intel Corporation
Nick Grobelny	Dell, Inc.
Patrick Debaenst	WISeKey Semiconductors
Patrick Gallo	United States Government
Paul England	Microsoft
Paul Stonelake	Micron Technology, Inc
Qi Huang	Advanced Micro Devices, Inc.
Rony Michaely	Lenovo (United States) INC
Silviu Vlasceanu	Huawei Technologies Co., Ltd.
Steve Clark	WISeKey Semiconductors
Subramanian Swaminathan	Juniper Networks, Inc.
Sven Schuch	Infineon Technologies
Theo Koulouris	Hewlett Packard Enterprise
Thomas Bowen	Intel Corporation
Thomas Deleuran	Huawei Technologies Co., Ltd.
Thomas Hardjono	MIT Connection Science
Thorsten Stremlau	NVIDIA Corporation
Todd Johnson	Cisco Systems
Tom Brostrom	Cyber Pack Ventures
Tom Dodson	Intel Corporation
Tom Laffey	Hewlett Packard Enterprise
Toru Tomita	NEC Corporation
Travis Gilbert	Dell, Inc.
Trevor Conn	Dell, Inc.
Vick Wei	Google Inc.
William Bellingrath	Juniper Networks, Inc.

Yucong Tao	Microsoft
Zachary Blum	United States Government
Zachary Halvorsen	Google Inc.
Zhang Li	Huawei Technologies Co., Ltd.
Zhoucan Gu	Google Inc.
Zhuo Liu	Huawei Technologies Co., Ltd.

CONTENTS

1 Scope and Context

Attestation is a critical element of a comprehensive endpoint assessment and integrity management capability. Attestation evidence generation is anticipated by several Trusted Computing use cases that rely on a Root of Trust, such as DICE and TPM. Attestation evidence is used by Verifiers to determine the state of a platform. Evidence is presented to a Verifier by an Attester. The Verifier must evaluate Evidence using assertions as a guide.

This specification complements information models for attestation (see section 1.3.1.2) by defining Reference Integrity Manifest (RIM) structures that a Verifier uses to validate expected values (Assertions) against actual values (Evidence). This RIM Information Model defines an abstract structure for assembling reference measurements (Assertions) that manufacturers and other supply chain entities assert as expected values.

A RIM information model has several characteristics. It:

- Identifies the creator (issuer) of the RIM instance.
- Identifies the supply chain entity that produces reference values.
- Contains reference measurements for installable software / firmware.
- Contains reference measurements for embedded firmware.
- Identifies the component, device or environment.
- Contains its own integrity protection capability (e.g., digital signature verification).
- Places constraints on RIM binding specifications that help ensure semantic interoperability and promote good security practice.

RIM binding specifications define a realization of RIM information model expressions. RIM binding specifications define formats, protocols, storage, and delivery methods used to instantiate and convey reference information to a Verifier. RIM binding may instantiate, store, and retrieve RIM data on an Attester's platform.

1.1 Audience

This document aids in the creation of binding specifications that define the formatting, structure, and usage guidelines for a given family of platforms. Verifier developers and platform developers working with RIM binding specifications may need to refer to this document for RIM element definitions.

1.2 Scope

The purpose of this specification is to:

1. Provide an information model that developers reference when writing RIM binding specifications.
2. Describe a common set of information elements that a Verifier supports:
    a. Cryptographic verification of the identity of the RIM structure creator.
    b. Cryptographic verification of RIM structures.
    c. Location of and access to RIM structures.
    d. Support for references to the RIM Binding specification used to realize a RIM structure.
3. Support a variety of platforms and devices that range from complex to simple; resource rich to resource constrained; server to embedded; cloud to IoT.

1.3 Relationships to Other Documents

1.3.1 TCG Specifications

There are several TCG specifications that define or use assertion data that may interest the reader.

1.3.1.1 RIMM

The Reference Integrity Measurement Manifest (RIMM) Schema specification [7] is a historical specification superseded by this specification. It is an XML schema representation of a reference manifest information model that contains elements and attributes. However, it pre-dates ISO-IE 19770-2 software ID (SWID), which has become increasingly popular.

1.3.1.2 TAP

TCG's Trusted Attestation Protocol (TAP) Information Model specification [1] defines the information elements used by Verifiers. Not all of the information elements are required by every Verifier. The RIM is essential for TAP based attestation. The RIM provides the assertions needed by the Verifier that implements the TAP model. Future versions of the TAP may include obtaining RIM information from the Attester.

1.3.1.3 FIM

The PC Client Firmware Integrity Measurement (FIM) specification [8] outlines the basic process for collecting, reporting, and processing (attestation) of PC Client firmware.

1.3.1.4 Platform Certificate

The TCG Platform Certificate Profile specification [10] contains assertions about trust made by a platform manufacturer. The certificate asserts the platform's security properties and configuration as shipped. The Platform Certificate Profile defines a PlatformConfigurationURI attribute that contains "URI where the reference integrity measurements could be obtained by the Verifier". Section 3.1.7 of the Platform Certificate Profile specification discusses options for the PlatformConfigurationURI attribute.

1.3.2 Non TCG Documents

This section identifies industry (non-TCG) information model specifications for manifest structures. The TCG RIM information model is patterned after this work.

1.3.2.1 NISTIR 8060

The National Institute for Standards and Technology Interagency Report (NISTIR) 8060 [3], "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags" is the primary reference for the elements described in this document. NIST IR 8060 uses definitions from ISO-IEC 19770-2, which is accessible on the NIST website. Because the TCG RIM Model specification is focused on integrity there are further restrictions and additional requirements for the information elements above and beyond the guidelines found in NISTIR 8060. Refer to section 4.1.1 for a discussion on NIST IR 8060 compliance. ISO-IEC 19770-2 (SWID).

ISO-IEC 19770-2 [4] International Organization for Standardization/International Electrotechnical Commission "Software identification tag" is known as the "SWID Specification" and is the main reference source for NIST IR 8060.

1.3.2.2 CoSWID

Concise Software Identifiers (CoSWID) [12] defines a concise representation of ISO/IEC 19770-2:2015 Software Identification (SWID) tags that are interoperable with the XML schema definition of ISO/IEC 19770-2:2015 and augmented for application in Constrained-Node Networks.

1.3.2.3 XML Signature Syntax and Processing

The XML Signature Syntax and Processing Version 2.0 [14] is an informative W3C Working Group Note that describes XML digital signature processing rules and syntax. XML Signatures provide integrity, message

authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere.

1.4 Terms and definitions

Asserter: A supply chain entity, manufacturer, vendor or reseller that produces reference values.

Assertions: Reference values.

Attester: Attester is defined by TAP [1] as “A platform or platform component that provides evidence to a Verifier as to its state”.

Base RIM: The Base RIM is a RIM instance that provides a verifiable identity of the RIM creator plus integrity information of support RIMs. The Base RIM contains a digest of each support RIM. The Base RIM also contains a signature.

Component RIM: A RIM bundle that applies to Firmware of a distinct make or model of a device that is integrated into a device that complies with the TCG PC Client RIM.

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

Endorsements: A protected statement from an entity (typically in a supply chain) that vouches for the trustworthiness of an Attester device**.**

Endorser: An entity that creates Endorsements that can be used to help evaluate the trustworthiness of Attesters.

Globally Unique ID (GUID): A GUID is referenced by ISO 19970-2 and is technically identical to the UUID specified by RFC 4122 [9].

Primary RIM Bundle: The first RIM Bundle in a RIM Bundle Collection.

Reference Integrity Manifest (RIM): A Reference Integrity Manifest contains structures that a Verifier uses to validate expected values (Assertions) against actual values (Evidence).

RIM Binding / Binding Specification: A specification that defines conventions for RIM (and RIM Bundle) formatting, marshalling, serialization, hashing, signing, encryption, realization, location, discovery or storage. A RIM Binding / Binding Specification also describes how the information contained in a RIM Bundle is transmitted between Attesters and Verifiers. For example, a RIM Bundle may be marshalled for conveyance over an IP-based communication protocol or instantiated as a file or collection of files in a file system.

RIM Bundle: A collection of RIM instances typically consisting of a single Base RIM and optional Support RIMs. A Bundle is created by a single entity at a single point in time.

RIM Bundle Collection: A collection of RIM Bundles typically consisting of a Primary RIM Bundle and one or more Supplemental RIM Bundles.

RIM Creator: Manufacturer, System Integrators, Value Added Resellers, Information Technology (IT) support organizations, or endpoint platform owners that create a RIM instance for an Endpoint platform.

RIM GUID: A GUID created as a reference to a specific RIM instance. This can be used to link a RIM instance to multiple other RIM instances.

RIM Instance: A RIM instance is a realization of the abstract RIM form according to a data structure binding method (see RIM Binding) where the resulting structure can be cryptographically hashed, signed or encrypted.

Supplemental RIM Bundle: Additional RIM Bundle added to a RIM Bundle Collection.

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

SWID: Software ID (SWID) tags defined by ISO-IEC 19770-2.

SWID Schema: An XML schema that describes the structure of the SWID tag.

Verifier: A system that analyzes evidence from an Attester to determine the Attester's state.

1.4.1 RIM Roles adopted from ISO 19770-2

The RIM information model adopts the roles defined by ISO 19770-2.

softwareCreator: Originator of the Software/Firmware as defined by ISO/IEC 19770-2.

licensor: Licensor of the Software/Firmware as defined by ISO/IEC 19770-2. Note that many OEMs license firmware from firmware developers and make modifications to the original firmware.

aggregator: An entity that encapsulates Software/Firmware from other organizations into their own distribution process. This implies that a System Integrator or Value-Added Reseller (VAR) will contribute supplemental RIM Bundles.

tagCreator: The default Role as defined by ISO/IEC 19770-2 for a primary RIM Bundle.

1.5 Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2]. This specification does not distinguish blocks of informative comments and normative requirements.

1.6 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statement.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind informative comment ... This is the second paragraph of text of the kind informative comment ... This is the nth paragraph of text of the kind informative comment ... To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 Background

Start of informative comment

The TCG TPM 2.0 Provisioning Guidance Document [6] describes a set of reference measurements that “represent the expected default values of the integrity measurements that the boot firmware and subsequent code generates and extends into TPM PCRs”. It further states that “Platform Manufacturers SHOULD deliver a list of expected integrity measurements of the platform BIOS, firmware, and other binaries they provide ‘as shipped’. Golden Measurements SHOULD be included in boot firmware updates, in order to support a given endpoint platforms lifecycle.”

The TCG Infrastructure Working Group Reference Manifest (RIMM) Schema Specification [7], defines an information model and data definition using XML Schema that offers a robust set of data items that could be used to generate a set of ‘Golden Measurements’. The TCG has since embraced several new technologies with options for formatting and content flexibility that better address the requirements of reference measurements. This specification supersedes the RIMM specification.

End of informative comment

3 Reference Integrity Manifest Information Model

The RIM Information Model consists of an abstract representation of elements and associated attributes that form a manifest called a Reference Integrity Manifest (RIM). The abstract representation is realized as a RIM Instance by applying a RIM Binding Specification that defines elements and attributes that are encoded for hashing, signing, transmission and storage. RIMs contain assertions about the state or configuration of a platform. Assertions contain values known to be correct according to the entity (vendor, organization, or person) who created or manufactured the platform or one of its hardware, firmware, or software elements.

3.1 RIM Bundle

There are two types of RIM, a Base RIM that can be digitally signed and optional Support RIMs that are referenced by the Base RIM.

If the PayloadType attribute of the Base RIM is set to “Direct” then the Payload Element within the Base RIM refers to reference measurements. In this case the payload items (File, Directories, Processes, or Resources) contain reference measurements that can be directly consumed by the Verifier as there is no Support RIM (as illustrated in Figure 1).

If the PayloadType attribute of the Base RIM is set to “Indirect”, the Payload Element within the Base RIM refers to one or more Support RIM as illustrated in Figure 2 (see section 4 for definition of the PayloadType attribute).

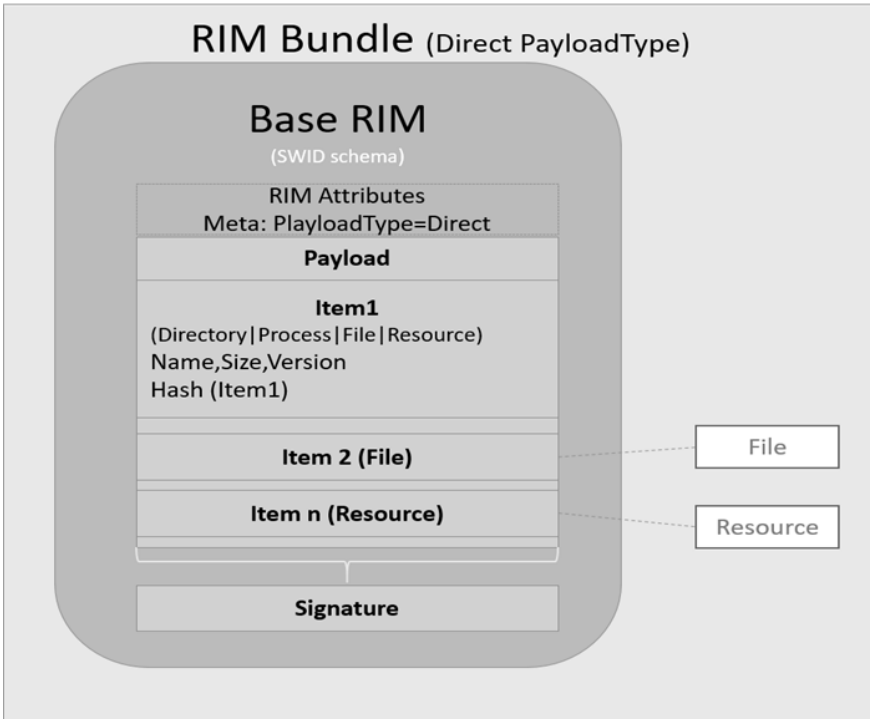

Figure 1: Figure 1 Structure of a RIM Bundle with a PayloadType set to Direct

If the PayloadType attribute of the Base RIM is set to “hybrid”, the Payload Element within the Base RIM refers to reference measurements that are individually labeled as Direct or Indirect using a Payload supportRimType attribute. See section 4.2 for details on the supportRimType attribute.

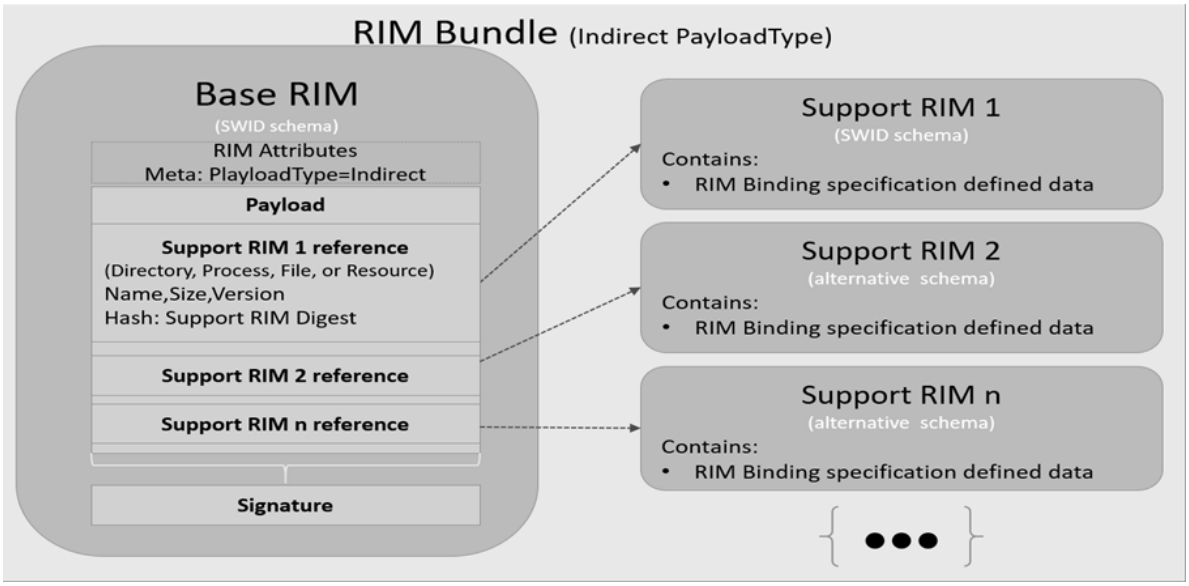

Figure 2 Structure of a RIM Bundle with a PayloadType set to Indirect

The Base RIM and optional Support RIMs are known as a RIM Bundle. Note that a Base RIM with a "Direct" Payload type is also considered a Bundle. A RIM Bundle with Support RIM(s) requires a RIM Binding specification that defines the formatting of the BASE RIM and the definition of the Support RIMs.

A platform's RIM Bundle Collection is a set of one or more RIM Bundles created by potentially multiple manufacturers, Value Added Resellers, or Maintenance Organizations (see section 3.1.3 for details on the RIM Bundle Collection).

3.1.1 Base Reference Integrity Manifest (Base RIM)

The Base RIM holds information either about Payloads that contain reference measurements or about Support RIMs that contain reference measurements. The Base RIM contains:

1. Cryptographically verifiable identification of the Creator of the RIM and Support RIMs.
2. A unique identifier (tagId) for a set of RIM Bundles.
3. An optional reference to the binding specification that defines the Support RIMs.
    a. Optional references to Support RIM, required if the Binding Specification is present
    b. Cryptographic hashes (digests) of all Payload references including Support RIMs.
4. A digital signature of the RIM signed by the RIM's Creator.

3.1.2 Support RIMs

Support RIMs are defined by a Binding RIM specification. The Binding RIM specification MUST define the following integrity measurements to be captured by the support RIMs:

1. Hash references (digests) for each critical Firmware/Software component that is to be evaluated by the Verifier.
2. Hash references (digests) for each critical event component that is to be evaluated by the Verifier.
3. Hash references (digests) for each critical Configuration Item that is to be evaluated by the Verifier

Note: The Support RIMs can be SWID files, log files, or custom files. The RIM Binding specification will define the file formats to be used.

3.1.3 RIM Bundle Collections

Any entity (e.g., Manufacturers, Value Added Resellers, or Maintenance Organizations) that changes or alters the existing reference measurements for a platform, can create a new RIM Bundle and make the new RIM Bundle available to a Verifier. The set of RIM Bundles created for the platform is referred to as a RIM Bundle Collection.

The lifecycle of a RIM Bundle Collection begins when an initial RIM Bundle is created for a given platform. The initial RIM bundle is referred to as the Primary RIM Bundle. A Supplemental RIM Bundle can be created for a given platform when another entity changes or modifies the platform in a manner that changes the contents of the existing RIM Bindle Collection.

Start of informative comment

As an example, when a platform moves through its lifecycle (see RIM Lifecycle in section 5 for details) there can be several different entities that participate in the production and configuration process(es). The Platform Supplier that places the Firmware on the platform should create the first RIM Bundle (RIM Bundle 1 in Figure 3).

End of informative comment

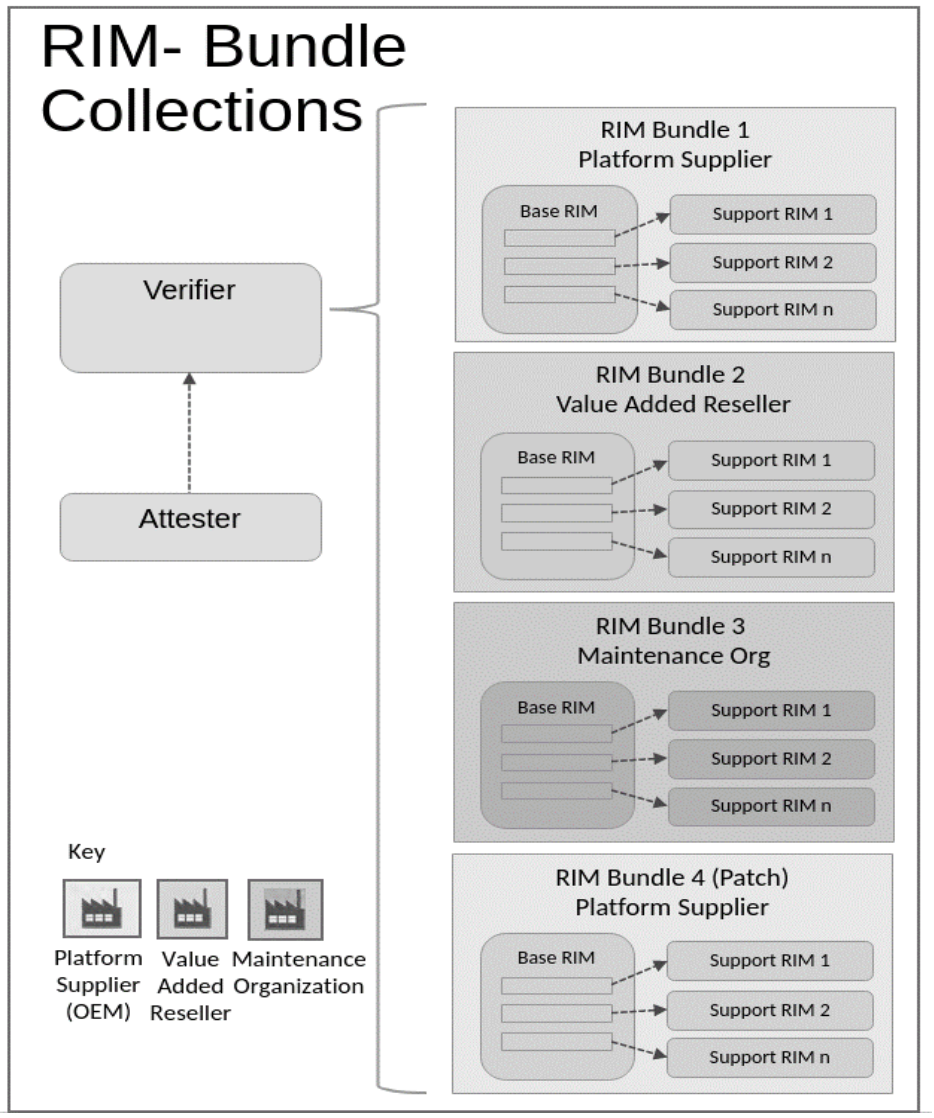

Figure 3 RIM Bundle Collection showing an evolution of changes to a device using a Base RIM with a Payload set to "Indirect".

Other entities create additional RIM Bundles (referred to as "Supplemental" bundles) and make them available to a Verifier. All RIM Bundles are linked by a set of RIM GUID references (the "tagId"). The Verifier MUST collect a set of RIM Bundles that reflect the production and maintenance of the platform. This specification refers to this collection as the platform's RIM Bundle Collection.

If a Value Added Reseller provides additional software that modifies the RIM (e.g., OS installation) then a Supplemental RIM Bundle is created (RIM Bundle 2).

A Maintenance Organization has an option to maintain RIM modifications. If the Maintenance Organization makes a configuration change that causes a modification to the existing RIM Bundles (e.g., it adds a BIOS password), then it would create a new RIM Bundle (see RIM Bundle 3 in Figure 3).

Each time a new patch is issued (typically issued by the Primary RIM Bundle creator) a new Patch Base RIM is created. (See RIM Bundle 4 in Figure 3).

The RIM Bundle Collection provides backward linkages using Base RIM tagIds that show the sequence of RIM Bundle additions. The sequence describes an evolution of platform changes. The linkage and sequencing can be useful to a Verifier.

3.2 Composite RIMs

Start of informative comment

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

The Composite RIM payload would include the PC Manufacturer Support RIM and a set of Base/Support RIMs for each component Manufacturer.

End of informative comment

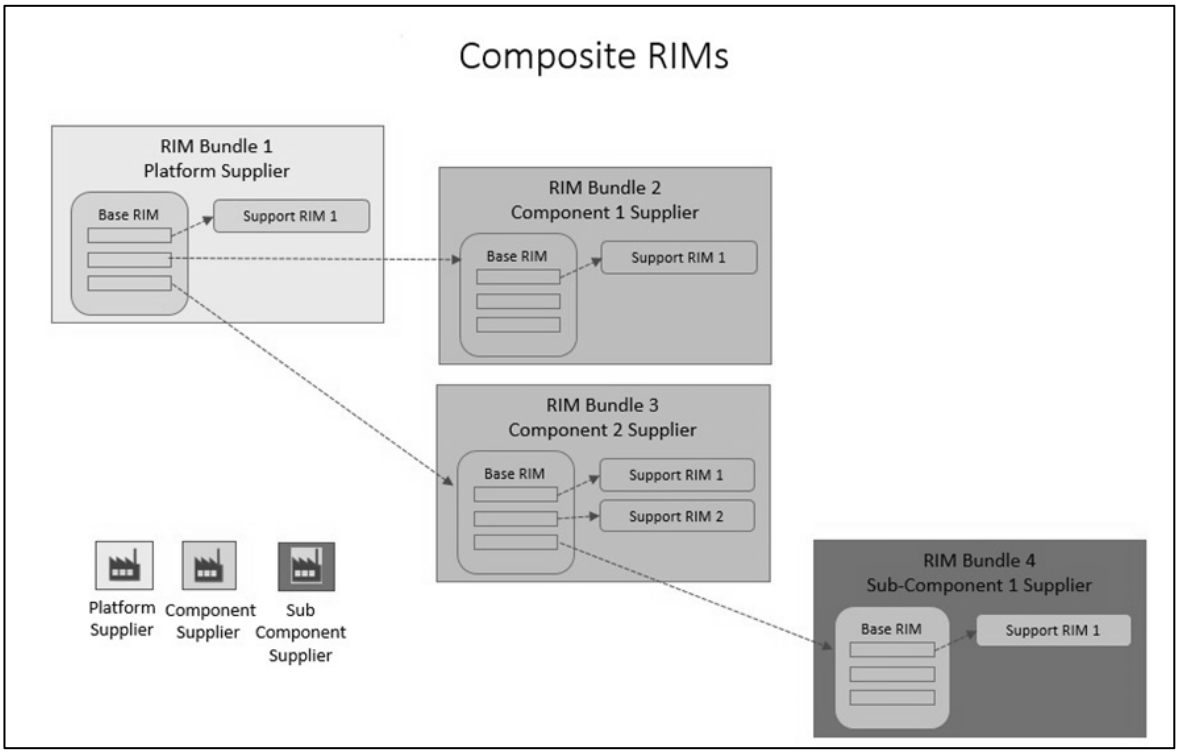

Figure 4 Composite RIM example showing Composite RIMs.

The Composite RIM in Figure 4 is defined as any RIM that references another Base RIM in its Payload Element. Attributes within the Payload element describe the location of the Base RIM reference and the Format of the Base RIM. Refer to section 4.2.1 for details on how a RIM Bundle can be embedded using the supportRimType and supportRimFormat attributes.

4 Base RIM Information Elements

The Base RIM contains a set of elements and associated attributes. The element and attribute definitions are defined in NIST IR 8660 and ISO/IEC19770-2, with the exception of the Meta elements defined in this section. NIST IR 8660 is the primary source for the definition of these elements. The RIM IM Element-Attribute table (Table 1) outlines the elements in the BASE RIM and the associated attributes. The Required column in Table 1 indicates whether the attribute is required in the implementation of the Base RIM.

NISTIR 8060 refers to Signature element as described in the World Wide Web Consortium (W3C) XML Signature Syntax and Processing (Second Edition) specification [14].

Element	NISTIR 8060 Reference	Attribute	Required	Notes
SoftwareIdentity	3.1.1	name	Required	MUST be the name of the product associated with this RIM. SHOULD be the Platform Model as defined in the Platform Certificate Specification [10]
		version	Required	MUST be the Version of the product associated with this RIM.
		versionScheme	Optional	As described in NISTIR 8060 section 5.1.2
		tagId	Required	MUST be a GUID.
		tagVersion	Required	MUST be an Integer, set by the tagCreator. This value SHOULD represent the version of patches issued.
		corpus	Optional	Not used for RIM, but MAY be used by an update package that installs a firmware update that includes an updated RIM. The default value is false.
		patch	Optional	MUST be set to false for the initial RIM. MUST be set to true for any RIM issued a subsequent time by a tagCreator. The default value is false.
		supplemental	Optional	MUST be set to false if the RIM is the first RIM to be created for the platform (e.g., it is a Primary RIM). MUST be set to true if the RIM is not the first RIM issued for the endpoint platform (typically issued by a System Integrator or Value Added Reseller). The default value is false.
Entity	3.1.2	name	Required	MUST be the name of the organization the created this RIM (the tagCreator).
		regid	Optional	SHOULD be a URI for the tag creator (e.g., www.example.com)
		role	Required	An extensible list of roles including “tag creator” and “aggregator”.
Link	3.1.4	href rel=”installation media”	Optional	SHOULD provide a link to obtain the item of type “install media”. Typically,

Element	NISTIR 8060 Reference	Attribute	Required	Notes
				a link to a download URL for the Firmware/Software installation package.
		href rel="supersedes" or rel="patches" or rel="requires"	Optional	SHOULD contain the tagId (GUID) of the previous RIM version (see NISTIR 8060 section 4.5.1). This can be used by Verifiers to reference the previous Base RIM. The Base RIM MUST include this attribute if the RIM refers to a patch. The associated rel attribute determines the usage
Meta	3.1.5	colloquialVersion	Optional	See section 4.1.1
		edition	Optional	See section 4.1.1
		product	Optional	See section 4.1.1
		revision	Optional	See section 4.1.1
		payloadType	Optional	See section 4.1.2
		platformManufacturerStr	Required	See section 4.1.3
		platformManufacturerId	Required	See section 4.1.3
		platformModel	Required	See section 4.1.3
		platformVersion	Optional	See section 4.1.3
		firmwareManufacturerStr	Optional	See section 4.1.3
		firmwareManufacturerId	Optional	See section 4.1.3
		firmwareModel	Optional	See section 4.1.3
		firmwareVersion	Optional	See section 4.1.3
		bindingSpec	Required	See section 4.1.4
		bindingSpecVersion	Required	See section 4.1.4
		pcUriLocal	Optional	See section 4.1.5
		pcUriGlobal	Optional	See section 4.1.5
		rimLinkHash	Optional	See section 4.1.6
Payload	3.1.6	file, directory, process, and/or resource	Required	MUST contain at least one or more references to a reference measurement if PayloadType="Direct" or to a Support RIM if PayloadType="Indirect". A Hash attribute is required for each reference. See section 4.1.2.
		name	Optional	MUST be present for File, Directory, and Process. MUST contain the name of the measurement target if PayloadType="Direct" or Support RIM if PayloadType="Indirect".
		size	Optional	MUST be present for File and Directory. SHOULD contain the size (in bytes) of the reference.
		hash	Required	MUST contain a Hash that holds the digest of the reference
		supportRimType	Optional	See section 4.2
		supportRimFormat	Optional	See section 4.2

Element	NISTIR 8060 Reference	Attribute	Required	Notes
		supportRimUriGlobal	Optional	See section 4.2
Signature	3.2	sigAlgorithm	Required	MUST be a TCG listed algorithm [16]
		hashAlgorithm	Optional	MUST be a TCG listed algorithm [16]
		keyInfoReference	Required	Must provide a reference to the certificate used to validate the Base RIM.
		digest	Optional	
		timestamp	Optional	
		signature	Required	

Table 1 RIM IM Element-Attribute table

4.1 Meta Element

NISTIR 8060 provides the following description: “Meta elements are used to record an array of optional metadata attributes related to the tag or the product”. A unique namespace, required for the Meta element, should use the prefix of “nTCGRim” and a URI of “https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model”.

4.1.1 NIST IR 8060 Required Attributes

Start of informative comment

Some attributes are required by NISTIR 8060 but are optional for Base RIM’s.

NIST IR 8060 defines a requirement for the versionScheme attribute but versionScheme is OPTIONAL for Base RIM’s. Refer to NIST IR 8060 for a definition of the versionScheme attribute.

NIST IR 8060 also defines required attributes for the Meta element:

“PRI-13. If appropriate values exist and can be determined, a <Meta> element MUST be provided and MUST furnish values for as many of the following attributes as possible: @product, @colloquialVersion, @revision, and @edition.”

NOTE: NIST IR 8060 implies that the requirement PRI-13 applies to primary tags, but PRI-13 is OPTIONAL for Base RIM’s.

The definitions of the @product, @colloquialVersion, @revision, and @edition attributes are in NIST IR 8060 and may be further refined by the RIM Binding specification.

End of informative comment

4.1.2 PayloadType

The Base RIM Payload may contain references that are used directly as reference measurements for a Verifier or it MAY refer to Support RIMs that contain the reference measurements. The possible values for PayloadType are:

Direct: The Base RIM Payload contains the reference integrity measurements. No Binding Specification is required for interpreting the payload contents.

Indirect: The Base RIM Payload contains the reference integrity measurements for Support RIM. A Binding Specification defines the Support RIM. This implies that Support RIM will be defined within the Binding Specification. The BindingSpec and BindingSpecVersion attribute are required for this Payload type.

Hybrid: The Base RIM Payload contains references that are a combination of Direct and Indirect. Each Payload item (Directory, File, or Resource) must have a supportRimType attribute as defined in section 4.2.

The default value for PayloadType is “Direct”.

4.1.3 RIM Platform Association

It may be useful for a Verifier to be able to associate a RIM with other TCG Artifacts such as a Platform Certificate or an Event Log (see Appendix C). It may also be useful to define a set of references that can be used to lookup the appropriate RIM from a database (which may hold thousands of RIMs) based upon fields within those artifacts. The following fields are found within all three artifacts:

4.1.3.1 platformManufacturerStr

The platformManufacturer is defined in the Platform Certificate specification as “Name of platform manufacturer as a string”. This Attribute MUST be present in the Meta element of the Base RIM.

4.1.3.2 platformManufacturerId

The PlatformManufacturerId is defined in the Platform Certificate specification as the “Platform manufacturer unique identifier as an IANA identifier [15]. This Attribute MUST be present in the Meta element of the Base RIM.

4.1.3.3 platformModel

The PlatformModel is defined in the Platform Certificate specification as a “Manufacturer-specific identifier”. This Attribute MUST be present in the Meta element of the Base RIM.

4.1.3.4 platformVersion

The PlatformVersion is defined in the Platform Certificate specification as a “Manufacturer-specific identifier”. This attribute MAY be present in the Meta element of the Base RIM.

4.1.3.5 firmwareManufacturerStr

The FirmwareManufacturerStr is the name of the firmware manufacturer as a string. This attribute does not need to be present in the Meta element of the Base RIM if the Firmware Manufacturer and the Platform Manufacturer are the same entity.

4.1.3.6 firmwareManufacturerId

The FirmwareManufacturerId is the IANA Private Enterprise Number [15] assigned to the Firmware manufacturer. This attribute does not need to be present in the Meta element of the Base RIM if the Firmware Manufacturer and the Platform Manufacturer are the same entity.

4.1.3.7 firmwareModel

The FirmwareModel is a string that specifies the manufacturer-specific Firmware model. This attribute MAY be present in the Meta element of the Base RIM.

4.1.3.8 firmwareVersion

The FirmwareVersion is a string that specifies manufacturer-specific firmware version value. This attribute MAY be present in the Meta element of the Base RIM.

4.1.4 bindingSpec attribute

If the Payload type is Indirect then the BindingSpec attribute SHOULD provide a reference to the Binding Specification to which the RIM bundle complies.

The BindingSpecVersion attribute holds the version of the Binding specification in a major.minor format.

4.1.5 pcUri attributes

The pcUri is the same as the PlatformConfigurationURI within the Platform Certificate. It provides a location of the information about the platform that SHOULD include any RIM Bundle created by the Platform Manufacturer. There are two options for the pcUri attribute:

- **pcUriLocal** is a URI where a copy of this RIM can be found within the platform itself.
- **pcUriGlobal** is a URI where a copy of this RIM can be found. This URI MUST match the PlatformConfigurationURI within the Platform Certificate.

4.1.6 rimLinkHash

The rimLinkHash is a base64 encoded SHA256 digest of the RIM referenced by the Link element (Href rel="supersedes" ,rel="patches, or rel="requires"). Primary Base RIM do not require rimLinkHash.

4.2 RIM IM defined Payload Attributes

ISO/IEC19770-2 allows for Directory, File, or Resource sub-elements to include an <any> attribute with <any> value. This RIM Information Model includes the definition of the attributes in Table 2.

ISO/IEC19770-2 Value	ISO/IEC19770-2 Type	ISO/IEC19770-2 Meaning
supportRimType	String	Set to either Direct or Indirect and applies to the specific Directory, File, or Resource within the Payload. Present only if the PayloadType is set to Hybrid. See section 4.2.1 for details.
supportRimFormat	String	Defined by the Binding Specification if more than one format of Support RIM is defined. See section 4.2.2 for details.
supportRimUriGlobal	String	The supportRimUriGlobal is a URI that holds a copy of the support RIM. See section 4.2.3 for details.

Table 2 RIM IM Defined Payload Attribute

4.2.1 supportRimType

Start of informative comment

The BaseRim supportRimType is intended to support the scenario discussed in section 3.2. The following illustration depicts the values associated with the supportRimType in the informative comment in section 3.2 scenario:

```
PC_BaseRIM
 |-------> Support RIM (supportRimType= Indirect)
 |-------> Component1_BaseRIM (supportRimType= BaseRim)
 |            |-------->Component1_Support RIM (supportRimType= Indirect)
 |-------> Component2_BaseRIM (supportRimType= BaseRim)
 |            |-------->Component2_Support RIM (supportRimType= Indirect)
 |            |-------->SubComponentA_BaseRIM (supportRimType= BaseRim)
 |                          |-------->SubComponentA_Support RIM (supportRimType= Indirect)
 End
```

End of informative comment

If the supportRimType attribute is used then the PayloadType Meta attribute MUST be set to “Hybrid”.

The supportRimType attribute MUST be a string set to “Direct”, “Indirect”, or “BaseRim”.

If set to “Direct” then the supportRimType attribute MUST contain a Reference Integrity Measurement in the hash field.

If set to “Indirect” then the supportRimType attribute MUST contain a reference to a support RIM along with a hash of the support RIM.

If set to “BaseRim” then the supportRimType MUST contain a reference to another Base RIM along with a hash of the Base RIM.

4.2.2 supportRimFormat

Start of informative comment

The supportRimFormat attribute is set by the tag creator and used by a Verifier if multiple formats of support RIM are defined by the specific RIM Binding specification (referenced by the BindingSpec and BindingSpecVersion attributes in table 1).

End of informative comment

The value of the supportRIMFormat attribute MUST be defined in the RIM Binding specification.

4.2.3 supportRimUriGlobal

Start of informative comment

The supportRimUriGlobal attribute is a URI that holds a copy of the support RIM. supportRimUriGlobal can be used to retrieve the support RIM should the support RIM become lost or damaged.

End of informative comment

The format of the supportRimUriGlobal attribute MUST be defined in the RIM Binding specification.

4.3 Signature

The Base RIM MUST have a single signature element signed by the tagCreator.

Start of informative comment

The Signature element constitutes a single “endorsement”.

It is helpful if the binding specification can include test sample(s) that illustrate how the signature element is applied.

The binding specification is responsible for defining the encoding of the Signature element and defining the various attributes contained within the signature element necessary for a verifier to be able to verify the signature.

End of informative comment

4.3.1 Layered Endorsements

Multiple entities may provide more than one independent signature on a single Base RIM.

The binding specification MAY include a description of how multiple signature elements from multiple parties can be applied to the Base Rim.

The addition of multiple signature elements is OPTIONAL and does not preclude the use of Base Rim signed by a single tagCreator.

4.3.2 Timestamps

A RIM signer may include a timestamp to note the time that the Base RIM was signed by the tagCreator.

The binding specification MAY provide a definition of a timestamp to be included as part of the Base RIM's signature.

Start of informative comment

There exist two timestamp scenarios of interest:

Countersignatures: Allowing a Trusted Third Party (TTP) authority to create a timestamp that can vouch for the fact that the RIM was signed at the specified time by the signer of the RIM. The countersignature is typically valid for a longer time than the RIM signature, which allows for validation of the RIM after the signer's certificate has expired.

Verifier RIM Policy: The Verifier may have policies that take into the account the time the RIM was created for determining the validity of the measurements with in the RIM.

End of informative comment

5 RIM Lifecycle

A RIM Lifecycle starts when Firmware/Software is initially installed on a platform, which is typically performed by a platform's manufacturer. System Integrators, Value Added Resellers, or IT Maintenance Organizations MAY provide Firmware/Software updates, change Firmware configuration, or make hardware changes that update the platform's RIM Bundle Collection. If they do, then they are responsible for creating new supplemental RIM Bundles.

5.1 Firmware/Software Installation

The first time that firmware/software is installed on the platform (typically at a manufacturing facility), the primary RIM Bundle is added to the RIM Bundle Collection (e.g., RIM Bundle 1 from Figure 3). This is the first RIM Bundle in the RIM Bundle Collection. The organization that creates the primary RIM Bundle takes on the role of tagCreator, defined in section 1.4.1. Installation typically requires the following sequence of actions:

1. A RIM GUID is created by the tagCreator to serve as a unique reference (tagId) within the RIM Bundle.
2. A RIM Bundle is created by the tagCreator.
    a. The Base RIM MUST contain a SoftwareIdentity element.
        i. The tagId attribute MUST be populated with the RIM GUID.
        ii. The tagVersion SHOULD be set to "0".
        iii. The patch attribute MUST be set to false.
        iv. The supplemental attribute MUST be set to false (indicating that this is a Primary Tag).
        v. The corpus attribute MUST be set to false.
    b. The Base RIM MUST contain an Entity element.
        i. Name MUST be the name of the tagCreator.
        ii. The Regid URI SHOULD be set to a URL that provides informative about the platform.
        iii. The Role SHOULD be set to tagCreator. Note that this not disallow a role of softwareCreator or Licensor depending upon the development model adopted by the tagCreator.
    c. The Base RIM SHOULD contain a Link element.
        i. The RIM MAY have a link to a download URL (rel=" install media") for firmware updates.
    d. The Base RIM MUST have a Meta element.
        i. The RIM MUST have a pcURI attribute that contains the PlatformConfigurationURI for the endpoint platform's Platform Certificate.
        ii. The BindingSpec and BindingSpecVersion attribute MUST be set to the RIM Binding specification that was used to build the RIM Bundle.
        iii. The PlatformManufacturerStr, PlatformManufacturerId, and PlatformModel MUST be populated. PlatformVersion MAY be populated.
        iv. FirmwareManufacturerStr, FirmwareManufacturerId, FirmwareModel, FirmwareVersion MAY be populated.
        v. The PayloadType attribute MAY be populated if the PayloadType is Indirect or hybrid.
        vi. pcUriLocal MAY be populated if the tagCreator supports access to the RIM on the platform.
        vii. pcUriGlobal MAY be populated if the tagCreator supports access to the RIM external to the platform.
    e. The Base RIM MUST contain a Payload element.
        i. The Payload element MUST have a File, Directory, Process, or Resource attribute.
            1. The attribute MUST have a name attribute that is specified in the RIM Binding specification.
            2. The attribute MUST have a size attribute that is specified in the RIM Binding specification.

3. The attribute MUST have a version attribute that is specified in the RIM Binding specification.
4. The attribute MAY have a Directory attribute that is specified in the RIM Binding specification.
5. The attribute MAY have supportRimType if the PayloadType is set to hybrid.
6. The attribute MAY have a supportRimFormat if the Binding Specification defined a supportRimFormat attribute.

ii. The Payload element Hash attribute MUST contain a digest of the Support RIM.

3. The tagCreator signs the Base RIM.
   a. The signature MUST contain a sigAlgorithm attribute that references a TCG listed algorithm [16] used to sign the Base RIM.
   b. The signature MUST contain a hashAlgorithm attribute that references a TCG listed algorithm [16] used to create a digest the Base RIM.
   c. The signature MUST contain a digest attribute that contains a cryptographic hash of the RIM used in the signature process.
   d. The signature MUST contain a KeyInfoReference attribute that references a certificate used by a Verifier to validate the signature on the Base RIM.
   e. The signature MUST contain a signature attribute that contains a cryptographic signature certificate used by a Verifier to validate the signature on the Base RIM.
4. The tagCreator distributes the RIM Bundle (which includes Supplemental RIM) as defined by the RIM Binding specification.

5.2 Pre-delivery Base RIM Modifications

If a System Integrator or Value Added Reseller make modifications to the platform that affect the measurement values (e.g., updating Firmware, adding Option ROMs, etc. to a platform) prior to the delivery of the platform to the end user, then that organization is responsible for creating a new supplemental RIM (e.g., RIM Bundle 2 from Figure 3). The System Integrator or Value Added Reseller takes on the role of aggregator, defined in section 1.4.1.

The aggregator is required to sign the supplemental RIM Bundle. The new (supplemental) RIM has the same requirements as the Base RIM created for the Firmware/Software installation (see section 5.1) with the following exceptions:

1. The SoftwareIdentity element supplemental attribute MUST be set to true.
2. The Link element Href attribute with rel=”supersedes” MUST be set to the previous Base RIM.
   a. The rimLinkHash MUST be populated with the digest of the previous Base RIM.
3. The Entity element Name MUST reflect the System Integrator/Value Added Reseller that is creating the new RIM Bundle.
4. The BindingSpec, BindingSpecVersion, PlatformManufacturerStr, PlatformManufacturerId, PlatformModel, and PlatformVersion MUST be set to the same values as the previous Base RIM.
5. FirmwareManufacturerStr, FirmwareManufacturerId, FirmwareModel FirmwareVersion MAY be set to a new value if the Firmware has been modified, otherwise it should be the same value as the previous Base RIM.
6. The Entity element Role MUST be set to aggregator.
7. The KeyInfoReference attribute that provides a reference to the signing certificate MUST reference the certificate used to validate the new supplemental RIM.
8. The Meta reference to the PlatformConfigurationURI MAY be updated. Note that a new Delta Platform Certificate, as defined in the Platform Certificate specification [10], MAY have to be created. If so, this Meta reference MUST be present.
9. The System Integrator/Value Added Reseller signs the Base RIM.

10. The aggregator distributes the RIM Bundle (which includes Supplemental RIM) as defined by the RIM Binding specification.

5.3 Maintenance updates

Maintenance Updates, as performed by an IT organization or a system owner MAY trigger a need for new RIM depending upon the nature of the update and the relationship between the items being measured (e.g., a Boot Order change may cause a TPM PCR change). In this case the IT organization may generate a new RIM and RIM Reference fields (e.g., RIM Bundle 3 from figure 3). The IT organization takes on the role of aggregator, defined in section 1.4.1. The aggregator is required to sign the supplemental RIM Bundle.

5.3.1 Creating a new (Maintenance) RIM

The new (supplemental) RIM has the same requirements as the RIM Bundle created for the Firmware/Software installation (see section 5.1) with the following exceptions:

1. The SoftwareIdentity element Supplemental attribute MUST be set to true.
2. The Entity element Name MUST reflect the new IT organization.
3. The Entity element Role MUST be set to aggregator.
4. The bindingSpec, bindingSpecVersion, platformManufacturerStr, platformManufacturerId, platformModel, and platformVersion, firmwareManufacturerStr, firmwareManufacturerId, firmwareModel, firmwareVersion MUST be set to the same values as the previous Base RIM.
5. The keyInfoReference attribute that provides a reference to the signing certificate MUST refer to the certificate used to validate the new supplemental RIM.
6. The Meta reference to the platformConfigurationUri MAY be updated. Note that if a Delta Platform Certificate (defined by the platform Certificate specification [10]) has been created, then this reference MUST be present.
7. The Maintenance Organization signs the new Base RIM.
8. The Maintenance Organization distributes the RIM bundle.

5.4 Firmware/Software Updates and Upgrades

Firmware/Software updates (aka. patches) will typically be created/distributed by the platform Manufacturer (e.g., the tagCreator). The Firmware updates can be a considered a patch or an upgrade (e.g., a new primary Base RIM). When a patch or upgrade occurs is at the discretion of the tagCreator.

5.4.1 Patches

For a patch, a new RIM Bundle is generated by the tagCreator (e.g., RIM Bundle 4 from Figure 3. The new (Patch) RIM follows the same requirements as the RIM Bundle created for the Firmware/Software installation (see section 5.1) with the following exceptions:

1. The SoftwareIdentity element tagVersion is set to the new Firmware/Software version.
2. A Link element is created with a href attribute that contains the tagId (GUID) of the previous Base RIM.
    a. A Link element Href attribute is created with a rel attribute that determines whether the update is a patch, required for another update, or supersedes a previous patch.
    b. The rimLinkHash MUST be populated with the digest of the previous Primary or Patch Base RIM.
3. The SoftwareIdentity element supplemental attribute MUST be set to false.
4. The SoftwareIdentity patch attribute MUST be set to true.
5. The Payload element MUST be updated to hold the digests of the new Support RIMs.

Additionally, the tagCreator MAY create a SoftwareIdentity element corpus SWID tag for Firmware/Software update package.

5.4.2 Upgrades

An upgrade requires a new Primary Base RIM and Support RIM (e.g., RIM Bundle 1 from Figure 3). The tagCreator MAY create a new tagId or reuse the old tagId (GUID). The new (Primary) RIM has the same requirements as the RIM Bundle created for the Firmware/Software installation (see section 5.1).

Appendix A: RIM Binding Specification Guidelines

Start of informative comment

In addition to the requirements detailed in this specification the Binding Specification should address the following requirements:

1. The Base RIM should be digitally signed using a TCG listed digital signing algorithm [16].
2. The binding specification should format the signature field of the Base RIM and define how signatures are to be applied.
3. The binding specification should define the value used for the BindingSpec attribute within the Meta field.
4. The Base and Supplemental RIMs should be formatted using XML (as described by NISTIR 8060) or CBOR (as described by the IEFT CoSWID specification [12]).
5. The Base RIM should hold a payload element for every support RIM Bundle specified in the Binding specification.
6. Delivery of the initial RIM Bundle to the Verifier should be addressed by the Binding specification. Delivery MAY include placement on the platform during manufacturing or a URI based retrieval mechanism. The LINK element in the Base RIM provides information about the location of updates and should be included in a RIM Bundle.
7. The RIM Binding specification should define how the Certificate Path used for the Validation of the RIM Bundle is to be obtained by a Verifier.
8. The binding specification must specify the use of a Platform Certificate and populate the PlatformConfigurationURI attribute.

End of informative comment

Appendix B: Direct Base RIM examples

Start of informative comment

All statements in this Appendix are Informative. It does not contain TCG Informative shading in order to retain readability.

End of informative comment

It is possible to use the SWID structure to define a Base RIM without a RIM Binding specification if the PayloadType attribute is set to “Direct”. These Appendix B examples describe an IOT platform with reference values for the platform’s Firmware. The example IOT platform has the following attributes:

Software Identity Name: IOTCore
  version: 01
  tagId: 94f6b457-9ac9-4d35-9b3f-78804173b651
  tagVersion:0
Entity (tagCreator) Name
  Regid: http://Example.com
  Role: softwareCreator tagCreator
Links:
  installation media url: https://Example.com/support/ProductA/firmware/installfiles
Meta:
  colloquialVersion: Firmware_2019
  edition: IOT
  product: ProductA
  revision: r2
  payloadType: Direct
  platformManufacturerStr: Example
  platformManufacturerId: 00201234
  platformModel: ProductA
  platformVersion:01
  firmwareManufacturerStr: FirmwareVendorA
  firmwareManufacturerId: 00213022
  firmwareVersion: 02
  bindingSpec: RIMIM
  bindingSpecVersion: 01
Payload:
  Directory: /boot/iot/iotBase/
  File1: Example.com.iotBase.bin
      version: 01.00
      size= 15400
      SHA256 hash: a314fc2dc663ae7a6b6bc6787594057396e6b3f569cd50fd5ddb4d1bbafd2b6a
  File2: iotExec.bin
      version: 01.00
      size: 1024
      SHA256 hash: 532eaabd9574880dbf76b9b8cc00832c20a6ec113d682299550d7a6e0f345e25

Example Using SWID

This section illustrates use of the SWID Payload element to construct a Base RIM that references the objects (files, directory, resources, or processes). The following example has been validated against the NIST Schema [18] using the NIST SWID validation tool [17].

The digest attribute of the Signature element contains a message digest over all the attributes within the RIM. The following is the SWID representation of the example platform's Base RIM:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <SoftwareIdentity
     xmlns="http://standards.iso.org/iso/19770/-2/2015/schema.xsd"
     xmlns:n8060="http://csrc.nist.gov/ns/swid/2015-extensions/1.0"
     xml:lang="en-US"
     supplemental="false"
     patch="false"
     corpus="false"
     tagVersion="0"
     tagId="94f6b457-9ac9-4d35-9b3f-78804173b651"
     version="01"
     versionScheme="alphanumeric"
     name="Example.com IOTCore" >
   <Entity
     name="Example Inc."
     role="softwareCreator tagCreator"
     regid="http://Example.com"/>
   <Link
     href="https://Example.com/support/ProductA/firmware/installfiles"
     rel="installationmedia"/>
<Meta xmlns:rim="https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model"
     colloquialVersion="Firmware_2019"
     edition="IOT"
     product="ProductA"
     revision="r2"
     rim:pcUriLocal="/boot/tcg/manifest/swidtag"
     rim:bindingSpec="IOT RIM"
     rim:bindingSpecVersion="1.2"
     rim:platformManufacturerId="00201234"
     rim:platformManufacturerStr="Example.com"
     rim:platformModel="ProductA"
     rim:firmwareManufacturer="BIOSVendorA"
     rim:firmwareManufacturerId="00213022"
     rim:rimLinkHash="88f21d8e44d4271149297404df91caf207130bfa116582408abd04ede6db7f51"/>
   <Payload xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
          n8060:envVarPrefix="$" n8060:envVarSuffix="" n8060:pathSeparator="/">
          <Directory
             name="iotBase"
             location="/boot/iot/">
```



```
            <File
              name="Example.com.iotBase.bin"
              version="01.00"
              size="15400"
              SHA256:hash="a314fc2dc663ae7a6b6bc6787594057396e6b3f569cd50fd5ddb4d1bbafd2b6a"/>
            <File
              name="iotExec.bin"
              version="01.00"
              size="1024"
              SHA256:hash="532eaabd9574880dbf76b9b8cc00832c20a6ec113d682299550d7a6e0f345e25"/>
          </Directory>
    </Payload>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ds:SignedInfo>
         <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
         <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha512"/>
         <ds:Reference URI="">
             <ds:Transforms>
               <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
             </ds:Transforms>
             <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
<ds:DigestValue>8A4C98225D855D388B42FD8AA4FA1646CFAC666E43FD6A7B950BB3E0B12134AA
</ds:DigestValue>
         </ds:Reference>
   </ds:SignedInfo>
<ds:SignatureValue>
        TSQUoVrQ0kg1eiltNwIhKPrIdsi1VhWjYNJlXvfQqW2EKk3X37X862SCfrz
        7v8IYJ7OorWwlFpGDStJDSR6saOScqSvmesCrGEEq+U6zegR9nH0lvcGZ8
        Rvc/y7U9kZrE4fHqEiLyfpmzJyPmWUT9Uta14nPJYsl3cmdThHB8Bs=
</ds:SignatureValue>
    <ds:KeyInfo>
       <ds:KeyName>
           "fd:a4:66:55:cf:7b:98:1c"
       </ds:KeyName>
       <ds11:KeyInfoReference xmlns:ds11="http://www.w3.org/2009/xmldsig11#">
           uri="https://Example.com/support/certificates/RIMSignerCert.pem"
        </ds11:KeyInfoReference>
   </ds:KeyInfo>
</ds:Signature>
</SoftwareIdentity>
```


Example Using CoSWID

A Base RIM can be encoded as a Concise Software Identity. The corresponding extension point $$coswid-extension is specified in IETF WGLC I-D.ietf-sacm-coswid-15. An additional map named reference-measurement-entry is added next to the software-meta-entry member in the root map of the CoSWID CDDL specification to represent Host Integrity at Runtime and Start-up (HIRS) Attestation [19]. The extension point is used as specified in section 2.1. in I-D.birkholz-rats-coswid-rim-01.

The CBOR encoding of CoSWID enables a compact representation of RIMs. The size of the SWID representation of the example platform's Base RIM is approximately 3000 Bytes after whitespace removal. The corresponding CoSWID representation is 884 Bytes - approximately 1/3 of the size of the SWID version. The CoSWID representation of the example platform's Base RIM shown below includes values found in the SWID version that are reused and converted from base64 representation to hex representation for illustrative purposes.

```
18( [ { 1: -39,
        3: "application/swid+cbor",
        "digest-algo-id": 1,
        "digest-value":
h'8A4C98225D855D388B42FD8AA4FA1646CFAC666E43FD6A7B950BB3E0B12134AA'
      },
      { 4: h'FDA46655CF7B981C',
        "key-info-reference":
"https://Example.com/support/certificates/RIMSignerCert.pem"
      },
      1398229316( { 15: "en-US",
                     0: "94f6b457-9ac9-4d35-9b3f-78804173b65as",
                    12: 0,
                     8: false,
                     9: false,
                    11: false,
                     1: "Example.com IOTCore",
                    13: "01",
                    14: 1,
                     5: { 45: "Firmware_2019",
                          47: "IOT",
                          52: "ProductA",
                          54: "r2"
                        },
                     2: { 31: "Example Inc.",
                          33: ["tag-creator", "software-creator"],
                          32: "http://Example.com"
                        },
                     4: { 38:
"https://Example.com/support/ProductA/firmware/installfiles",
                          40: 4
                        },
                     6: { 16: { 23: "/boot/iot/",
                                24: "iotBase",
                                26: { 17: [ { 24: "Example.com.iotBase.bin",
                                              20: 15400,
                                              21: "01.00",
                                               7: [ 1,

h'A314FC2DC663AE7A6B6BC6787594057396E6B3F569CD50FD5DDB4D1BBAFD2B6A'
                                                  ]
                                          },
```



```
                                          { 24: "iotExec.bin",
                                            20: 1024,
                                            21: "01.00",
                                             7: [ 1,

h'532EAABD9574880DBF76B9B8CC00832C20A6EC113D682299550D7A6E0F345E25'
                                                 ]
                                          }
                                        ]
                                    }
                                }
                            },
                        58: { 62: "/boot/tcg/manifest/swidtag",
                              63: "IOT RIM",
                              64: "1.2",
                              65: 201234,
                              66: "Example.com",
                              67: "ProductA",
                              70: "BIOSVendorA",
                              69: 213022,
                              73:
 h'88F21D8E44D4271149297404DF91CAF207130BFA116582408ABD04EDE6DB7F51'
                            }
                    }

),h'4D2414A15AD0D248357A296D37022128FAC876C8B55615A360D2655EF7D0A96D842A4DD7DFB5FCEB64827
EBCFBBFC21827B3A8AD6C251691834AD243491EAC68E49CA92BE67AC0AB18412AF94EB37A047D9C7D25BDC199
F11BDCFF2ED4F6466B1387C7A8488BC9FA66CC9C8F996513F54B5AD789CF258B25DDC99D4E11C1F01B'
    ]
  )
```


Appendix C: RIM References to other TCG defined objects

A RIM is referenced by the TCG Event Log and the TCG Platform Certificate. The TCG Platform Certificate has several fields that map between the TCG Event Log and the Base RIM. These include the fields and attributes in Table 3:

Platform Certificate fields	TCG Event Log Event Sp8000-155Event Attribute	RIM IM Attribute
PlatformManufacturerId	PlatformManufacturerId	platformManufacturerId
PlatformManufacturerStr	PlatformManufacturerStr	platformManufacturerStr
PlatformModel	PlatformModel	platformModel
PlatformVersion	N/A	platformVersion
componentClass (Firmware)	N/A	N/A
componentManufacturer	FirmwareManufacturerStr	firmwareManufacturerStr
componentManufacturerId	FirmwareManufacturerId	firmwareManufacturerId
componentRevision	FirmwareRevsion	firmwareRevision
PlatformConfiguratioURI	N/A	pcURI

Table 3: TCG defined objects to RIM IM Attribute cross references

The TCG Platform Certificate Profile [10] also defines a PlatformConfigurationURI attribute that contains “URI where the reference integrity measurements could be obtained by the verifier”. The Platform Certificates Platform Configuration URI MUST hold a reference to the location of the Primary RIM Bundle. The URI MAY point to a location on the Endpoint platform or it MAY reside in a publicly accessible location. The Base RIM MUST contain a LINK element that points to the same URI as the Platform Certificate’s PlatformConfigurationURI.

The RIM Binding Specification MAY define a location for the Platform Certificate. It may reside on the TPM, on the endpoint platform, or anywhere a URI can point.

The TCG Event log as defined by the PC Client Firmware profile [13] has a specific event, referred to as the TCG_Sp800-155_Event, that can be used by a verifier to match the RIM Bundle collection to a specific appraisal request. The ReferenceManifestGuid (see Figure 4 RIM References) should match the tagId found within each of the RIM Bundles that pertain to the Firmware.

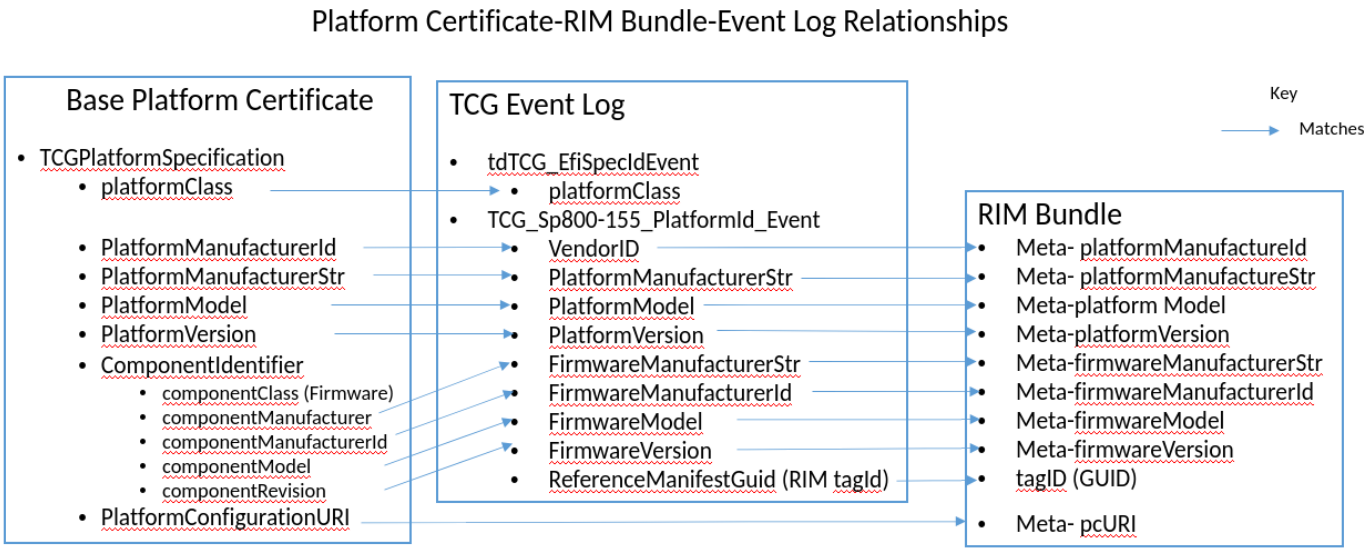

Figure 4 RIM References

Appendix D: References

[1] Trusted Computing Group, “Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0”, https://trustedcomputinggroup.org/wp-content/uploads/TNC_TAP_Information_Model_v1.00_r0.29A_publicreview.pdf

[2] IEFT RFC-2119, "Key words for use in RFCs to Indicate Requirement Levels", https://tools.ietf.org/html/rfc2119

[3] NISTIR-8660, “Guidelines for the Creation of Interoperable Software ID (SWID) Tags”, April 2016, https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf

[4] ISO/IEC 19770-2:2015 International Organization for Standardization/International Electrotechnical Commission, Information technology -- Software asset management -- Part 2: Software identification tag, ISO/IEC 19770-2:2009, November 2009. http://www.iso.org/iso/catalogue_detail?csnumber=65666

[5] H. Birkholz “"A SUIT Manifest Extension for Concise Software Identifiers draft-birkholz-suit-coswid-manifest-00", IETF , January 2019, https://tools.ietf.org/html/draft-birkholz-suit-coswid-manifest-00

[6] TCG TPM 2.0 Provisioning Guidance, March 2017, https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf

[7] TCG Infrastructure Working Group Reference Manifest (RIMM) Schema Specification, November 2006, https://trustedcomputinggroup.org/wp-content/uploads/IWG-Reference_Manifest_Schema_Specification_v1.pdf

[8] TCG PC Client Platform Firmware Integrity Measurement (FIM) specification, https://trustedcomputinggroup.org/resource/tcg-pc-client-platform-firmware-integrity-measurement/

[9] RFC 4122 A Universally Unique IDentifier (UUID) URN Namespace, https://tools.ietf.org/html/rfc4122

[10] TCG Platform Certificate Profile, Version 1.1 Revision 1 5 13 Feb 2019, https://trustedcomputinggroup.org/wp-content/uploads/IWG_Platform_Certificate_Profile_v1p1_r15_pubrev.pdf

[11] The Cryptographic Algorithm Validation Program (CAVP). https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program

[12] H. Birkholz “Concise Software Identifiers draft-ietf-sacm-coswid-08,” November 2018, IETF https://www.rfc-editor.org/rfc/rfc9393.html

[13] Trusted Computing Group, “TCG PC Client Platform Firmware Profile Specification” https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClientSpecPlat_TPM_2p0_1p04_pub.pdf

[14] W3C Working Group Note, "XML Signature Syntax and Processing Version 2.0", 23 July 2015, https://www.w3.org/TR/xmldsig-core2/

[15] IANA Private Enterprise Numbers, http://www.iana.org/assignments/enterprise-numbers/enterprise-numbers

[16] TCG Algorithm Registry, Family “2.0", Level 00 Revision 01.22,February 9, 2015,https://trustedcomputinggroup.org/wp-content/uploads/TCG_Algorithm_Registry_Rev_1.22.pdf

[17] NIST, The SWID Tag Validation (SWIDVal) Tool Version 0.5.0 , https://csrc.nist.gov/CSRC/media/Projects/Software-Identification-SWID/tools/swidval-0.5.0-swidval.zip

[18] NIST SWID Tag extensions from NIST IR 8060, https://csrc.nist.gov/schema/swid/2015-extensions/swid-2015-extensions-1.0.xsd

[19] Host Integrity at Runtime and Start-up (HIRS), https://github.com/nsacyber/hirs

