RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2020

H. Birkholz
M. Eckel
Fraunhofer SIT
S. Bhandari
B. Sulzen
E. Voit
Cisco
L. Xia
Huawei
T. Laffey
HPE
G. Fedorkow
Juniper

March 11, 2020

A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs

draft-ietf-rats-yang-tpm-charra-01

Abstract

This document defines a YANG RPC and a minimal datastore tree required to retrieve attestation evidence about integrity measurements from a composite device with one or more roots of trust for reporting. Complementary measurement logs are also provided by the YANG RPC originating from one or more roots of trust of measurement. The module defined requires at least one TPM 1.2 or TPM 2.0 and corresponding Trusted Software Stack included in the device components of the composite device the YANG server is running on.

Table of Contents

1. Introduction

This document is based on the terminology defined in the [I-D.ietf-rats-architecture] and uses the interaction model and information elements defined in the [I-D.birkholz-rats-reference-interaction-model] document. The currently supported hardware security modules (HWM) - sometimes also referred to as an embedded secure element (eSE) - is the Trusted Platform Module (TPM) version 1.2 and 2.0 specified by the Trusted Computing Group (TCG). One ore more TPMs embedded in the components of a composite device - sometimes also referred to as an aggregate device - are required in order to use the YANG module defined in this document. A TPM is used as a root of trust for reporting (RTR) in order to retrieve attestation evidence from a composite device (quote primitive operation). Additionally, it is used as a root of trust for storage (RTS) in order to retain shielded secrets and store

system measurements using a folding hash function (extend primitive operation).

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The YANG Module for Basic Remote Attestation Procedures

One or more TPMs MUST be embedded in the composite device that is providing attestation evidence via the YANG module defined in this document. The ietf-basic-remote-attestation YANG module enables a composite device to take on the role of Claimant and Attester in accordance with the Remote Attestation Procedures (RATS) architecture [I-D.ietf-rats-architecture] and the corresponding challenge-response interaction model defined in the [I-D.birkholz-rats-reference-interaction-model] document. A fresh nonce with an appropriate amount of entropy MUST be supplied by the YANG client in order to enable a proof-of-freshness with respect to the attestation evidence provided by the attester running the YANG datastore. The functions of this YANG module are restricted to 0-1 TPMs per hardware component.

2.1. Tree Diagram

```
module: ietf-tpm-remote-attestation
  +--ro rats-support-structures
     +--ro supported-algos*   uint16
     +--ro compute-nodes* [node-id]
        +--ro node-id                string
        +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
        +--ro node-name?             string
        +--ro node-location?         string
        +--ro tpms* [tpm-name]
           +--ro tpm-name                     string
           +--ro tpm-physical-index?          int32 {ietfhw:entity-mib}?
           +--ro tpm-manufacturer?            string
           +--ro tpm-firmware-version?        string
           +--ro tpm-specification-version?   string
           +--ro tpm-status?                  string
           +--ro certificates* []
              +--ro certificate
                 +--ro certificate-name?       string
                 +--ro certificate-type?       enumeration
```

```
               +--ro certificate-value?      ietfct:end-entity-cert-cms
               +--ro lak-public-structure?   binary

  rpcs:
    +---x tpm12-challenge-response-attestation
    |  +---w input
    |  |  +---w tpm1-attestation-challenge
    |  |     +---w pcr-indices*            uint8
    |  |     +---w nonce-value             binary
    |  |     +---w TPM_SIG_SCHEME-value    uint8
    |  |     +---w (key-identifier)?
    |  |     |  +--:(public-key)
    |  |     |  |  +---w pub-key-id?       binary
    |  |     |  +--:(TSS_UUID)
    |  |     |     +---w TSS_UUID-value
    |  |     |        +---w ulTimeLow?       uint32
    |  |     |        +---w usTimeMid?       uint16
    |  |     |        +---w usTimeHigh?      uint16
    |  |     |        +---w bClockSeqHigh?   uint8
    |  |     |        +---w bClockSeqLow?    uint8
    |  |     |        +---w rgbNode*         uint8
    |  |     +---w add-version?            boolean
    |  |     +---w tpm-name?               string
    |  |     +---w tpm-physical-index?     int32 {ietfhw:entity-mib}?
    |  +--ro output
    |     +--ro tpm12-attestation-response* [tpm-name]
    |        +--ro tpm-name                     string
    |        +--ro tpm-physical-index?          int32 {ietfhw:entity-mib}?
    |        +--ro up-time?                     uint32
    |        +--ro node-id?                     string
    |        +--ro node-physical-index?         int32 {ietfhw:entity-mib}?
    |        +--ro fixed?                       binary
    |        +--ro external-data?               binary
    |        +--ro signature-size?              uint32
    |        +--ro signature?                   binary
    |        +--ro (tpm12-quote)
    |           +--:(tpm12-quote1)
    |           |  +--ro version* []
    |           |  |  +--ro major?      uint8
    |           |  |  +--ro minor?      uint8
    |           |  |  +--ro revMajor?   uint8
    |           |  |  +--ro revMinor?   uint8
    |           |  +--ro digest-value?          binary
    |           |  +--ro TPM_PCR_COMPOSITE* []
    |           |     +--ro pcr-indices*       uint8
    |           |     +--ro value-size?        uint32
    |           |     +--ro tpm12-pcr-value*   binary
    |           +--:(tpm12-quote2)
```

```
    |              +--ro tag?                   uint8
    |              +--ro pcr-indices*           uint8
    |              +--ro locality-at-release?   uint8
    |              +--ro digest-at-release?     binary
    +---x tpm20-challenge-response-attestation
    |  +---w input
    |  |  +---w tpm20-attestation-challenge
    |  |     +---w nonce-value          binary
    |  |     +---w challenge-objects* [node-id tpm-name]
    |  |        +---w node-id                       string
    |  |        +---w node-physical-index?          int32 {ietfhw:entity-mib}?
    |  |        +---w tpm-name                      string
    |  |        +---w tpm-physical-index?           int32 {ietfhw:entity-mib}?
    |  |        +---w pcr-list* []
    |  |        |  +---w pcr
    |  |        |     +---w pcr-indices*                  uint8
    |  |        |     +---w (algo-registry-type)
    |  |        |        +--:(tcg)
    |  |        |        |  +---w tcg-hash-algo-id?       uint16
    |  |        |        +--:(ietf)
    |  |        |           +---w ietf-ni-hash-algo-id?   uint8
    |  |        +---w (signature-identifier-type)
    |  |        |  +--:(TPM_ALG_ID)
    |  |        |  |  +---w TPM_ALG_ID-value?       uint16
    |  |        |  +--:(COSE_Algorithm)
    |  |        |     +---w COSE_Algorithm-value?   int32
    |  |        +---w (key-identifier)?
    |  |           +--:(public-key)
    |  |           |  +---w pub-key-id?             binary
    |  |           +--:(uuid)
    |  |              +---w uuid-value?             binary
    |  +--ro output
    |     +--ro tpm20-attestation-response* [node-id tpm-name]
    |        +--ro tpm-name                    string
    |        +--ro tpm-physical-index?         int32 {ietfhw:entity-mib}?
    |        +--ro up-time?                    uint32
    |        +--ro node-id                     string
    |        +--ro node-physical-index?        int32 {ietfhw:entity-mib}?
    |        +--ro quote?                      binary
    |        +--ro quote-signature?            binary
    |        +--ro pcr-bank-values* [algo-registry-type]
    |        |  +--ro (algo-registry-type)
    |        |  |  +--:(tcg)
    |        |  |  |  +--ro tcg-hash-algo-id?       uint16
    |        |  |  +--:(ietf)
    |        |  |     +--ro ietf-ni-hash-algo-id?   uint8
    |        |  +--ro pcr-values* [pcr-index]
    |        |     +--ro pcr-index    uint16
```

```
|       |     +--ro pcr-value?   binary
|       +--ro pcr-digest-algo-in-quote
|          +--ro (algo-registry-type)
|             +--:(tcg)
|             |  +--ro tcg-hash-algo-id?       uint16
|             +--:(ietf)
|                +--ro ietf-ni-hash-algo-id?   uint8
+---x basic-trust-establishment
|  +---w input
|  |  +---w nonce-value                   binary
|  |  +---w (signature-identifier-type)
|  |  |  +--:(TPM_ALG_ID)
|  |  |  |  +---w TPM_ALG_ID-value?      uint16
|  |  |  +--:(COSE_Algorithm)
|  |  |     +---w COSE_Algorithm-value?   int32
|  |  +---w tpm-name?                     string
|  |  +---w tpm-physical-index?           int32 {ietfhw:entity-mib}?
|  |  +---w certificate-name?             string
|  +--ro output
|     +--ro attestation-certificates* [tpm-name]
|        +--ro tpm-name                   string
|        +--ro tpm-physical-index?        int32 {ietfhw:entity-mib}?
|        +--ro up-time?                   uint32
|        +--ro node-id?                   string
|        +--ro node-physical-index?       int32 {ietfhw:entity-mib}?
|        +--ro certificate-name?          string
|        +--ro attestation-certificate?   ietfct:end-entity-cert-cms
|        +--ro (key-identifier)?
|           +--:(public-key)
|           |  +--ro pub-key-id?          binary
|           +--:(uuid)
|              +--ro uuid-value?          binary
+---x log-retrieval
   +---w input
   |  +---w log-selector* [node-id tpm-name]
   |  |  +---w node-id                    string
   |  |  +---w node-physical-index?       int32 {ietfhw:entity-mib}?
   |  |  +---w tpm-name                   string
   |  |  +---w tpm-physical-index?        int32 {ietfhw:entity-mib}?
   |  |  +---w (index-type)?
   |  |  |  +--:(last-entry)
   |  |  |  |  +---w last-entry-value?    binary
   |  |  |  +--:(index)
   |  |  |  |  +---w last-index-number?   uint64
   |  |  |  +--:(timestamp)
   |  |  |     +---w timestamp?           yang:date-and-time
   |  |  +---w log-entry-quantity?        uint16
   |  |  +---w pcr-list* []
```

```
       |  |     +---w pcr
       |  |        +---w pcr-indices*                 uint8
       |  |        +---w (algo-registry-type)
       |  |           +--:(tcg)
       |  |           |  +---w tcg-hash-algo-id?      uint16
       |  |           +--:(ietf)
       |  |              +---w ietf-ni-hash-algo-id?   uint8
       |  +---w log-type        identityref
      +--ro output
         +--ro system-event-logs
            +--ro node-data* [node-id tpm-name]
               +--ro node-id                string
               +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
               +--ro up-time?               uint32
               +--ro tpm-name               string
               +--ro tpm-physical-index?    int32 {ietfhw:entity-mib}?
               +--ro log-result
                  +--ro (log-type)
                     +--:(bios)
                     |  +--ro bios-event-logs
                     |     +--ro bios-event-entry* [event-number]
                     |        +--ro event-number    uint32
                     |        +--ro event-type?     uint32
                     |        +--ro pcr-index?      uint16
                     |        +--ro digest-list* []
                     |        |  +--ro (algo-registry-type)
                     |        |  |  +--:(tcg)
                     |        |  |  |  +--ro tcg-hash-algo-id?       uint16
                     |        |  |  +--:(ietf)
                     |        |  |     +--ro ietf-ni-hash-algo-id?   uint8
                     |        |  +--ro digest*                       binary
                     |        +--ro event-size?     uint32
                     |        +--ro event-data*     uint8
                     +--:(ima)
                        +--ro ima-event-logs
                           +--ro ima-event-entry* [event-number]
                              +--ro event-number              uint64
                              +--ro ima-template?             string
                              +--ro filename-hint?            string
                              +--ro filedata-hash?            binary
                              +--ro filedata-hash-algorithm?  string
                              +--ro template-hash-algorithm?  string
                              +--ro template-hash?            binary
                              +--ro pcr-index?                uint16
                              +--ro signature?                binary
```

2.2. YANG Module

This YANG module imports modules from [RFC6991], [RFC8348], and [I-D.ietf-netconf-crypto-types].

```
<CODE BEGINS> file ietf-tpm-remote-attestation@2019-01-07.yang
module ietf-tpm-remote-attestation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-tpm-remote-attestation";
  prefix "yang-rats-charra";

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-hardware {
      prefix ietfhw;
  }
  import ietf-crypto-types {
      prefix ietfct;
  }

  organization
   "IETF RATS (Remote ATtestation procedureS) Working Group";

  contact
   "WG Web  : <http://datatracker.ietf.org/wg/rats/>
    WG List : <mailto:rats@ietf.org>
    Author  : Henk Birkholz <henk.birkholz@sit.fraunhofer.de>
    Author  : Michael Eckel <michael.eckel@sit.fraunhofer.de>
    Author  : Shwetha Bhandari <shwethab@cisco.com>
    Author  : Bill Sulzen <bsulzen@cisco.com>
    Author  : Eric Voit <evoit@cisco.com>
    Author  : Liang Xia (Frank) <frank.xialiang@huawei.com>
    Author  : Tom Laffey <tom.laffey@hpe.com>
    Author  : Guy Fedorkow <gfedorkow@juniper.net>";

  description
    "A YANG module to enable a TPM 1.2 and TPM 2.0 based
     remote attestation procedure using a challenge-response
     interaction model and the TPM 1.2 and TPM 2.0 Quote
     primitive operations.

     Copyright (c) 2020 IETF Trust and the persons identified
     as authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
```

```
    Legal Provisions Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX
    (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
    itself for full legal notices.

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here.";

  revision "2020-03-09" {
    description
      "Initial version";
    reference
      "draft-ietf-rats-yang-tpm-charra";
  }

  /*****************/
  /*   Groupings   */
  /*****************/

  grouping hash-algo {
    description
      "A selector for the hashing algorithm";
    choice algo-registry-type {
      mandatory true;
      description
        "Unfortunately, both IETF and TCG have registries here.
         Choose your weapon wisely.";
      case tcg {
        description
          "You chose the east door, the tcg space opens up to
           you.";
        leaf tcg-hash-algo-id {
          type uint16;
          description
            "This is an index referencing the TCG Algorithm
            Registry based on TPM_ALG_ID.";
        }
      }
      case ietf {
        description
          "You chose the west door, the ietf space opens up to
           you.";
```

```
        leaf ietf-ni-hash-algo-id {
          type uint8;
          description
            "This is an index referencing the Named Information
             Hash Algorithm Registry.";
        }
      }
    }
  }

  grouping hash {
    description
      "The hash value including hash-algo identifier";
    list hash-digests {
      description
        "The list of hashes.";
      container hash-digest {
        description
          "A hash value based on a hash algorithm registered by an
           SDO.";
        uses hash-algo;
        leaf hash-value {
          type binary;
          description
            "The binary representation of the hash value.";
        }
      }
    }
  }

  grouping nonce {
    description
      "A nonce to show freshness and counter replays.";
    leaf nonce-value {
      type binary;
      mandatory true;
      description
        "This nonce SHOULD be generated via a registered
         cryptographic-strength algorithm. In consequence,
         the length of the nonce depends on the hash algorithm
         used. The algorithm used in this case is independent
         from the hash algorithm used to create the hash-value
         in the response of the attestor.";
    }
  }

  grouping tpm12-pcr-selection {
    description
```

```
      "A Verifier can request one or more PCR values using its
       individually created Attestation Key Certificate (AC).
       The corresponding selection filter is represented in this
       grouping.
       Requesting a PCR value that is not in scope of the AC used,
       detailed exposure via error msg should be avoided.";
    leaf-list pcr-indices {
      type uint8;
      description
        "The numbers/indexes of the PCRs. At the moment this is limited
        to 32.";
    }
  }

  grouping tpm20-pcr-selection {
    description
      "A Verifier can request one or more PCR values uses its
       individually created AC. The corresponding selection filter is
       represented in this grouping. Requesting a PCR value that is not
       in scope of the AC used, detailed exposure via error msg should
       be avoided.";
    list pcr-list {
      description
        "For each PCR in this list an individual list of banks
         (hash-algo) can be requested. It depends on the datastore, if
         every bank in this grouping is included per PCR (crude), or if
         each requested bank set is returned for each PCR individually
         (elegant).";
      container pcr {
        description
          "The composite of a PCR number and corresponding bank
          numbers.";
        leaf-list pcr-indices {
           type uint8;
           description
             "The number of the PCR. At the moment this is limited
             32";
        }
        uses hash-algo;
      }
    }
  }

  grouping pcr-selector {
    description
      "A Verifier can request the generation of an attestation
       certificate (a signed public attestation key
       (non-migratable, tpm-resident) wrt one or more PCR values.
```

```
       The corresponding creation input is represented in this grouping.
       Requesting a PCR value that is not supported results in an error,
       detailed exposure via error msg should be avoided.";
    list pcr-list {
      description
        "For each PCR in this list an individual hash-algo can be
         requested.";
      container pcr {
        description
          "The composite of a PCR number and corresponding bank
           numbers.";
        leaf-list pcr-index {
           type uint8;
           description
             "The numbers of the PCRs that are associated with
              the created key. At the moment the highest number is 32";
        }
        uses hash-algo;
      }
    }
  }

  grouping tpm12-signature-scheme {
    description
      "The signature scheme used to sign the evidence via a TPM 1.2.";
    leaf TPM_SIG_SCHEME-value {
      type uint8;
      mandatory true;
      description
        "Selects the signature scheme that is used to sign the TPM
         Quote information response. Allowed values can be found in
         the table at the bottom of page 32 in the TPM 1.2 Structures
         specification (Level 2 Revision 116, 1 March 2011).";
    }
  }

  grouping tpm20-signature-scheme {
    description
      "The signature scheme used to sign the evidence.";
    choice signature-identifier-type {
      mandatory true;
      description
        "There are multiple ways to reference a signature type.
         This used to select the signature algo to sign the quote
         information response.";
      case TPM_ALG_ID {
        description
          "This references the indices of table 9 in the TPM 2.0
```

```
           structure specification.";
        leaf TPM_ALG_ID-value {
          type uint16;
          description
            "The TCG Algorithm Registry ID value.";
        }
      }
      case COSE_Algorithm {
        description
          "This references the IANA COSE Algorithms Registry indices.
           Every index of this registry to be used must be mapable to a
           TPM_ALG_ID value.";
        leaf COSE_Algorithm-value {
          type int32;
          description
            "The IANA COSE Algorithms ID value.";
        }
      }
    }
  }

  grouping tpm12-attestation-key-identifier {
    description
      "A selector for a suitable key identifier for a TPM 1.2.";
    choice key-identifier {
      description
        "Identifier for the attestation key to use for signing
         attestation evidence.";
      case public-key {
        leaf pub-key-id {
          type binary;
          description
            "The value of the identifier for the public key.";
        }
      }
      case TSS_UUID {
        description
          "Use a YANG agent generated (and maintained) attestation
           key UUID that complies with the TSS_UUID datatype of the TCG
           Software Stack (TSS) Specification, Version 1.10 Golden,
           August 20, 2003.";
        container TSS_UUID-value {
          description
            "A detailed structure that is used to create the
             TPM 1.2 native TSS_UUID as defined in the TCG Software
             Stack (TSS) Specification, Version 1.10 Golden,
             August 20, 2003.";
          leaf ulTimeLow {
```

```
            type uint32;
            description
              "The low field of the timestamp.";
          }
          leaf usTimeMid {
            type uint16;
            description
              "The middle field of the timestamp.";
          }
          leaf usTimeHigh {
            type uint16;
            description
              "The high field of the timestamp multiplexed with the
               version number.";
          }
          leaf bClockSeqHigh {
            type uint8;
            description
              "The high field of the clock sequence multiplexed with
               the variant.";
          }
          leaf bClockSeqLow {
            type uint8;
            description
              "The low field of the clock sequence.";
          }
          leaf-list rgbNode {
            type uint8;
            description
              "The spatially unique node identifier.";
          }
        }
      }
    }
  }

  grouping tpm20-attestation-key-identifier {
    description
      "A selector for a suitable key identifier.";
    choice key-identifier {
      description
        "Identifier for the attestation key to use for signing
         attestation evidence.";
      case public-key {
        leaf pub-key-id {
          type binary;
          description
            "The value of the identifier for the public key.";
```

```
        }
      }
      case uuid {
        description
          "Use a YANG agent generated (and maintained) attestation
           key UUID.";
        leaf uuid-value {
          type binary;
          description
            "The UUID identifying the corresponding public key.";
        }
      }
    }
  }

  grouping tpm-identifier {
    description
      "In a system with multiple-TPMs get the data from a specific TPM
       identified by the name and physical-index.";
    leaf tpm-name {
      type string;
      description
        "Name value of a single TPM or 'All'";
    }
    leaf tpm-physical-index {
      if-feature ietfhw:entity-mib;
      type int32 {
        range "1..2147483647";
      }
      config false;
      description
        "The entPhysicalIndex for the TPM.";
      reference
        "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
  }
  grouping compute-node-identifier {
    description
      "In a distributed system with multiple compute nodes
       this is the node identified by name and physical-index.";
    leaf node-id {
       type string;
       description
         "ID of the compute node, such as Board Serial Number.";
    }
    leaf node-physical-index {
      if-feature ietfhw:entity-mib;
      type int32 {
```

```
        range "1..2147483647";
      }
      config false;
       description
         "The entPhysicalIndex for the compute node.";
       reference
         "RFC 6933: Entity MIB (Version 4) - entPhysicalIndex";
    }
  }

  grouping tpm12-pcr-info-short {
    description
      "This structure is for defining a digest at release when the only
       information that is necessary is the release configuration.";
    uses tpm12-pcr-selection;
    leaf locality-at-release {
      type uint8;
      description
        "This SHALL be the locality modifier required to release the
         information (TPM 1.2 type TPM_LOCALITY_SELECTION)";
    }
    leaf digest-at-release {
      type binary;
        description
          "This SHALL be the digest of the PCR indices and PCR values
           to verify when revealing auth data (TPM 1.2 type
           TPM_COMPOSITE_HASH).";
    }
  }

  grouping tpm12-version {
    description
      "This structure provides information relative the version of
      the TPM.";
    list version {
      description
        "This indicates the version of the structure
         (TPM 1.2 type TPM_STRUCT_VER). This MUST be 1.1.0.0.";
      leaf major {
        type uint8;
        description
          "Indicates the major version of the structure.
           MUST be 0x01.";
      }
      leaf minor {
        type uint8;
        description
          "Indicates the minor version of the structure.
```

```
          MUST be 0x01.";
      }
      leaf revMajor {
        type uint8;
        description
          "Indicates the rev major version of the structure.
           MUST be 0x00.";
      }
      leaf revMinor {
        type uint8;
        description
          "Indicates the rev minor version of the structure.
           MUST be 0x00.";
      }
    }
  }

  grouping tpm12-quote-info-common {
    description
      "These statements are used in bot quote variants of the TPM 1.2";
    leaf fixed {
      type binary;
      description
        "This SHALL always be the string 'QUOT' or 'QUO2'
         (length is 4 bytes).";
    }
    leaf external-data {
      type binary;
      description
        "160 bits of externally supplied data, typically a nonce.";
    }
    leaf signature-size {
      type uint32;
      description
       "The size of TPM 1.2 'signature' value.";
    }
    leaf signature {
      type binary;
      description
        "Signature over SHA-1 hash of tpm12-quote-info2'.";
    }
  }

  grouping tpm12-quote-info {
    description
      "This structure provides the mechanism for the TPM to quote the
       current values of a list of PCRs (as used by the TPM_Quote2
       command).";
```

```
    uses tpm12-version;
    leaf digest-value {
      type binary;
      description
        "This SHALL be the result of the composite hash algorithm using
         the current values of the requested PCR indices
         (TPM 1.2 type TPM_COMPOSITE_HASH.)";
    }
  }

  grouping tpm12-quote-info2 {
    description
      "This structure provides the mechanism for the TPM to quote the
       current values of a list of PCRs
       (as used by the TPM_Quote2 command).";
    leaf tag {
      type uint8;
      description
        "This SHALL be TPM_TAG_QUOTE_INFO2.";
    }
    uses tpm12-pcr-info-short;
  }

  grouping tpm12-cap-version-info {
    description
      "TPM returns the current version and revision of the TPM 1.2 .";
    list TPM_PCR_COMPOSITE {
      description
        "The TPM 1.2 TPM_PCRVALUEs for the pcr-indices.";
      uses tpm12-pcr-selection;
      leaf value-size {
        type uint32;
        description
          "This SHALL be the size of the 'tpm12-pcr-value' field
           (not the number of PCRs).";
      }
      leaf-list tpm12-pcr-value {
        type binary;
        description
          "The list of TPM_PCRVALUEs from each PCR selected in sequence
           of tpm12-pcr-selection.";
      }
      list version-info {
        description
          "An optional output parameter from a TPM 1.2 TPM_Quote2.";
        leaf tag {
          type uint16;
          description
```

```
            "The TPM 1.2 version and revision
             (TPM 1.2 type TPM_STRUCTURE_TAG).
             This MUST be TPM_CAP_VERSION_INFO (0x0030)";
        }
        uses tpm12-version;
        leaf spec-level {
          type uint16;
          description
            "A number indicating the level of ordinals supported.";
        }
        leaf errata-rev {
          type uint8;
          description
            "A number indicating the errata version of the
             specification.";
        }
        leaf tpm-vendor-id {
          type binary;
          description
            "The vendor ID unique to each TPM manufacturer.";
        }
        leaf vendor-specific-size {
          type uint16;
          description
            "The size of the vendor-specific area.";
        }
        leaf vendor-specific {
          type binary;
          description
            "Vendor specific information.";
        }
      }
    }
  }

  grouping tpm12-pcr-composite {
    description
      "The actual values of the selected PCRs (a list of TPM_PCRVALUEs
       (binary) and associated metadata for TPM 1.2.";
    list TPM_PCR_COMPOSITE {
      description
        "The TPM 1.2 TPM_PCRVALUEs for the pcr-indices.";
      uses tpm12-pcr-selection;
      leaf value-size {
        type uint32;
        description
          "This SHALL be the size of the 'tpm12-pcr-value' field
           (not the number of PCRs).";
```

```
      }
      leaf-list tpm12-pcr-value {
        type binary;
        description
          "The list of TPM_PCRVALUEs from each PCR selected in sequence
           of tpm12-pcr-selection.";
      }
    }
  }

  grouping node-uptime {
    description
      "Uptime in seconds of the node.";
    leaf up-time {
      type uint32;
      description
        "Uptime in seconds of this node reporting its data";
    }
  }

  identity log-type {
    description
      "The type of logs available.";
  }

  identity bios {
    base log-type;
    description
      "Measurement log created by the BIOS/UEFI.";
  }

  identity ima {
    base log-type;
    description
      "Measurement log created by IMA.";
  }

  grouping log-identifier {
    description
      "Identifier for type of log to be retrieved.";
    leaf log-type {
      type identityref {
        base log-type;
      }
      mandatory true;
      description
        "The corresponding measurement log type identity.";
    }
```

```
  }

  grouping boot-event-log {
    description
      "Defines an event log corresponding to the event that extended the
       PCR";
    leaf event-number {
      type uint32;
      description
        "Unique event number of this event";
    }
    leaf event-type {
        type uint32;
        description
          "log event type";
    }
    leaf pcr-index {
      type uint16;
      description
        "Defines the PCR index that this event extended";
    }
    list digest-list {
      description "Hash of event data";
      uses hash-algo;
      leaf-list digest {
        type binary;
        description
          "The hash of the event data";
      }
    }
    leaf event-size {
      type uint32;
      description
        "Size of the event data";
    }
    leaf-list event-data {
      type uint8;
      description
        "The event data size determined by event-size";
    }
  }

  grouping ima-event {
    description
      "Defines an hash log extend event for IMA measurements";
    leaf event-number {
      type uint64;
      description
```

```
        "Unique number for this event for sequencing";
    }
    leaf ima-template {
      type string;
      description
        "Name of the template used for event logs
         for e.g. ima, ima-ng, ima-sig";
    }
    leaf filename-hint {
      type string;
      description
        "File that was measured";
    }
    leaf filedata-hash {
      type binary;
      description
        "Hash of filedata";
    }
    leaf filedata-hash-algorithm {
      type string;
      description
        "Algorithm used for filedata-hash";
    }
    leaf template-hash-algorithm {
      type string;
      description
        "Algorithm used for template-hash";
    }
    leaf template-hash {
      type binary;
      description
        "hash(filedata-hash, filename-hint)";
    }
    leaf pcr-index {
      type uint16;
      description
        "Defines the PCR index that this event extended";
    }
    leaf signature {
      type binary;
      description
        "The file signature";
    }
  }

  grouping bios-event-log {
    description
      "Measurement log created by the BIOS/UEFI.";
```

```
      list bios-event-entry {
     key event-number;
       description
         "Ordered list of TCG described event log
          that extended the PCRs in the order they
          were logged";
     uses boot-event-log;
   }
 }

 grouping ima-event-log {
   list ima-event-entry {
     key event-number;
     description
     "Ordered list of ima event logs by event-number";
      uses ima-event;
   }
   description
     "Measurement log created by IMA.";
 }

 grouping event-logs {
   description
     "A selector for the log and its type.";
   choice log-type {
     mandatory true;
     description
       "Event log type determines the event logs content.";
     case bios {
       description
         "BIOS/UEFI event logs";
       container bios-event-logs {
         description
           "This is an index referencing the TCG Algorithm
            Registry based on TPM_ALG_ID.";
         uses bios-event-log;
       }
     }
     case ima {
       description
         "IMA event logs";
       container ima-event-logs {
         description
           "This is an index referencing the TCG Algorithm
            Registry based on TPM_ALG_ID.";
         uses ima-event-log;
       }
     }
```

```
    }
  }

  /**********************/
  /*   RPC operations   */
  /**********************/

  rpc tpm12-challenge-response-attestation {
    description
      "This RPC accepts the input for TSS TPM 1.2 commands of the
       managed device. ComponentIndex from the hardware manager YANG
       module to refer to dedicated TPM in composite devices,
       e.g. smart NICs, is still a TODO.";
    input {
      container tpm1-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 1.2 structure definitions";
        uses tpm12-pcr-selection;
        uses nonce;
        uses tpm12-signature-scheme;
        uses tpm12-attestation-key-identifier;
        leaf add-version {
          type boolean;
          description
            "Whether or not to include TPM_CAP_VERSION_INFO; if true,
             then TPM_Quote2 must be used to create the response.";
        }
        uses tpm-identifier;
      }
    }
    output {
      list tpm12-attestation-response {
        key tpm-name;
        description
          "The binary output of TPM 1.2 TPM_Quote/TPM_Quote2, including
           the PCR selection and other associated attestation evidence
           metadata";
        uses tpm-identifier;
        uses node-uptime;
        uses compute-node-identifier;
        uses tpm12-quote-info-common;
        choice tpm12-quote {
          mandatory true;
          description
            "Either a tpm12-quote-info or tpm12-quote-info2, depending
```

```
            on whether TPM_Quote or TPM_Quote2 was used
            (cf. input field add-verson).";
          case tpm12-quote1 {
            description
              "BIOS/UEFI event logs";
            uses tpm12-quote-info;
            uses tpm12-pcr-composite;
          }
          case tpm12-quote2 {
            description
              "BIOS/UEFI event logs";
            uses tpm12-quote-info2;
          }
        }
      }
    }
  }

  rpc tpm20-challenge-response-attestation {
    description
      "This RPC accepts the input for TSS TPM 2.0 commands of the
       managed device. ComponentIndex from the hardware manager YANG
       module to refer to dedicated TPM in composite devices,
       e.g. smart NICs, is still a TODO.";
    input {
      container tpm20-attestation-challenge {
        description
          "This container includes every information element defined
           in the reference challenge-response interaction model for
           remote attestation. Corresponding values are based on
           TPM 2.0 structure definitions";
        uses nonce;
        list challenge-objects {
          key "node-id tpm-name";
          description
            "Nodes to fetch attestation information, PCR selection
             and AK identifier.";
          uses compute-node-identifier;
          uses tpm-identifier;
          uses tpm20-pcr-selection;
          uses tpm20-signature-scheme;
          uses tpm20-attestation-key-identifier;
        }
      }
    }
    output {
      list tpm20-attestation-response {
        key "node-id tpm-name";
```

```
        description
          "The binary output of TPM2b_Quote in one TPM chip of the
           node which identified by node-id. An TPMS_ATTEST structure
           including a length, encapsulated in a signature";
        uses tpm-identifier;
        uses node-uptime;
        uses compute-node-identifier;
        leaf quote {
          type binary;
          description
            "Quote data returned by TPM Quote, including PCR selection,
             PCR digest and etc.";
        }
        leaf quote-signature {
          type binary;
          description
            "Quote signature returned by TPM Quote.";
        }
        list pcr-bank-values {
          key algo-registry-type;
          description
            "PCR values in each PCR bank.";
             uses hash-algo;
          list pcr-values {
            key pcr-index;
            description
              "List of one PCR bank.";
            leaf pcr-index {
              type uint16;
              description
                "PCR index number.";
            }
            leaf pcr-value {
              type binary;
              description
                "PCR value.";
            }
          }
        }
        container pcr-digest-algo-in-quote {
          uses hash-algo;
          description
            "The hash algorithm for PCR value digest in
             Quote output.";
        }
      }
    }
  }
```

```
  rpc basic-trust-establishment {
    description
      "This RPC creates a tpm-resident, non-migratable key to be used
       in TPM_Quote commands, an attestation certificate.";
    input {
      uses nonce;
      uses tpm20-signature-scheme;
      uses tpm-identifier;
      leaf certificate-name {
         type string;
         description
           "An arbitrary name for the identity certificate chain
            requested.";
      }
    }
    output {
      list attestation-certificates {
        key tpm-name;
        description
          "Attestation Certificate data from a TPM identified by the TPM
           name";
        uses tpm-identifier;
        uses node-uptime;
        uses compute-node-identifier;
        leaf certificate-name {
          type string;
          description
            "An arbitrary name for this identity certificate or
             certificate chain.";
        }
        leaf attestation-certificate {
          type ietfct:end-entity-cert-cms;
          description
            "The binary signed certificate chain data for this identity
             certificate.";
        }
        uses tpm20-attestation-key-identifier;
      }
    }
  }

  rpc log-retrieval {
    description
      "Logs Entries are either identified via indices or via providing
       the last line received. The number of lines returned can be
       limited. The type of log is a choice that can be augmented.";
    input {
      list log-selector {
```

```
        key "node-id tpm-name";
        description
          "Selection of log entries to be reported.";
        uses compute-node-identifier;
        uses tpm-identifier;
        choice index-type {
          description
            "Last log entry received, log index number, or timestamp.";
          case last-entry {
            description
              "The last entry of the log already retrieved.";
            leaf last-entry-value {
              type binary;
              description
                "Content of an log event which matches 1:1 with a
                 unique event record contained within the log.  Log
                 entries subsequent to this will be passed to the
                 requester.  Note: if log entry values are not unique,
                 this MUST return an error.";
            }
          }
          case index {
            description
              "Numeric index of the last log entry retrieved, or zero.";
            leaf last-index-number {
              type uint64;
              description
                "The last numeric index number of a log entry.
                 Zero means to start at the beginning of the log.
                 Entries subsequent to this will be passed to the
                 requester.";
            }
          }
          case timestamp {
            leaf timestamp {
              type yang:date-and-time;
              description
                "Timestamp from which to start the extraction.  The next
                 log entry subsequent to this timestamp is to be sent.";
            }
            description
              "Timestamp from which to start the extraction.";
          }
        }
        leaf log-entry-quantity {
          type uint16;
          description
            "The number of log entries to be returned. If omitted, it
```

```
            means all of them.";
        }
        uses tpm20-pcr-selection;
      }
      uses log-identifier;
    }

    output {
      container system-event-logs {
        description
          "The requested data of the measurement event logs";
        list node-data {
          key "node-id tpm-name";
          description
            "Event logs of a node in a distributed system
             identified by the node name";
          uses compute-node-identifier;
          uses node-uptime;
          uses tpm-identifier;
          container log-result {
            description
              "The requested entries of the corresponding log.";
               uses event-logs;
          }
        }
      }
    }
  }

  /*********************************/
  /*   Protocol accessible nodes   */
  /*********************************/

  container rats-support-structures {
    config false;
    description
      "The datastore definition enabling verifiers or relying
       parties to discover the information necessary to use the
       remote attestation RPCs appropriately.";
    leaf-list supported-algos {
      type uint16;
      description
        "Supported TPM_ALG_ID values for the TPM in question.
         Will include ComponentIndex soon.";
    }
    list compute-nodes {
      key node-id;
      uses compute-node-identifier;
```

```
      description
        "A list names of hardware componnets in this composite
         device that RATS can be conducted with.";
      leaf node-name {
        type string;
        description
          "Name of the compute node.";
      }
      leaf node-location {
        type string;
        description
          "Location of the compute node, such as slot number.";
      }
      list tpms {
        key tpm-name;
        uses tpm-identifier;
        description
          "A list of TPMs in this composite device that RATS
           can be conducted with.";
        leaf tpm-manufacturer {
          type string;
          description
            "TPM manufacturer name.";
        }
        leaf tpm-firmware-version {
          type string;
          description
            "TPM firmware version.";
        }
        leaf tpm-specification-version {
          type string;
          description
            "TPM1.2 or TPM2.0.";
        }
        leaf tpm-status {
          type string;
          description
            "TPM chip self-test status, normal or abnormal.";
        }
        list certificates {
          description
            "The TPM's certificates, including EK certificates
             and AK certificates.";
          container certificate {
            description
              "Three types of certificates can be accessed via
               this statement, including Initial Attestation
               Key Cert, Local Attestation Key Cert or
```

```
               Endorsement Key Cert.";
            leaf certificate-name {
               type string;
               description
                 "An arbitrary name for this identity certificate
                  or certificate chain.";
            }
            leaf certificate-type {
              type enumeration {
                enum endorsement-cert {
                  value 0;
                  description
                    "EK Cert type.";
                }
                enum initial-attestation-cert {
                  value 1;
                  description
                    "IAK Cert type.";
                }
                enum local-attestation-cert {
                  value 2;
                  description
                    "LAK Cert type.";
                }
              }
              description
                "Type of this certificate";
            }
            leaf certificate-value {
              type ietfct:end-entity-cert-cms;
              description
                "The binary signed public endorsement key (EK),
                 attestation key(AK) and corresponding claims
                 (EK,AK Certificate). In a TPM 2.0 the EK,
                 AK Certificate resides in a well-defined NVRAM
                 location by the TPM vendor. Maybe certificate-value
                 defined as binary type is a simple way.";
            }
            leaf lak-public-structure {
              type binary;
              description
                "Marshalled LAK public structure, used for LAK
                 Certificate verification";
            }
          }
        }
      }
    }
```

```
  }
}
<CODE ENDS>
```

3. IANA considerations

This document will include requests to IANA:

To be defined yet.

4. Security Considerations

There are always some.

5. Acknowledgements

Not yet.

6. Change Log

Changes from version 00 to version 01:

- Addressed author's comments
- Extended complementary details about attestation-certificates
- Relabeled chunk-size to log-entry-quantity
- Relabeled location with compute-node or tpm-name where appropriate
- Added a valid entity-mib physical-index to compute-node and tpm-name to map it back to hardware inventory
- Relabeled name to tpm_name
- Removed event-string in last-entry

7. References

7.1. Normative References

[I-D.birkholz-rats-reference-interaction-model]
Birkholz, H. and M. Eckel, "Reference Interaction Models for Remote Attestation Procedures", draft-birkholz-rats-reference-interaction-model-02 (work in progress), January 2020.

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-14 (work in progress), March 2020.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

7.2. Informative References

[I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote Attestation Procedures Architecture", draft-ietf-rats-architecture-02 (work in progress), March 2020.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Michael Eckel
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: michael.eckel@sit.fraunhofer.de

Shwetha Bhandari
Cisco Systems

Email: shwethab@cisco.com

Bill Sulzen
Cisco Systems

Email: bsulzen@cisco.com

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Liang Xia (Frank)
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

Tom Laffey
Hewlett Packard Enterprise

Email: tom.laffey@hpe.com

Guy C. Fedorkow
Juniper Networks
10 Technology Park Drive
Westford, Massachusetts 01886

Email: gfedorkow@juniper.net