RATS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

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

July 08, 2019

YANG Module for Basic Challenge-Response-based Remote Attestation Procedures
draft-birkholz-rats-basic-yang-module-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 a TPM 2.0 and corresponding Trusted Software Stack included in the device components of the composite device the YANG server is running on.

Status of This Memo

Table of Contents

1. Introduction

This document is based on the terminology defined in the [I-D.birkholz-attestation-terminology] and uses the interaction model and information elements defined in the [I-D.birkholz-rats-reference-interaction-model] document. The currently supported hardware security module (HWM) - sometimes also referred to as an embedded secure element(eSE) - is the Trusted Platform Module (TPM) 2.0 specified by the Trusted Computing Group (TCG). One ore more TPM 2.0 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 2.0 is used as a root of trust for reporting (RTR) in order to retrieve attestation evidence from a composite device. Additionally, it is used as a root of trust for measurement (RTM) in order to provide event logs - sometimes also referred to as measurement logs.

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 RFC 2119, BCP 14 [RFC2119].

2. The YANG Module for Basic Remote Attestation Procedures

One or more TPM 2.0 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.birkholz-attestation-terminology] 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 TPM 2.0 per hardware component.

2.1. Tree format

```
<CODE BEGINS>
module: ietf-basic-remote-attestation
  +--ro rats-support-structures
     +--ro supported-algos*   uint16
     +--ro tpms* [tpm_name]
     |  +--ro tpm_name              string
     |  +--ro tpm-physical-index?   int32 {ietfhw:entity-mib}?
     |  +--ro certificates* []
     |     +--ro certificate
     |        +--ro certificate-name?    string
     |        +--ro certificate-type?    enumeration
     |        +--ro certificate-value?   ietfct:end-entity-cert-cms
     +--ro compute-nodes* [node-name]
        +--ro node-name              string
        +--ro node-physical-index?   int32 {ietfhw:entity-mib}?

  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-name?                   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 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 nonce-value                   binary
    |  |  |  +---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
    |  |  +---w tpms* [tpm_name]
    |  |     +---w tpm_name              string
    |  |     +---w tpm-physical-index?   int32 {ietfhw:entity-mib}?
    |  +--ro output
    |     +--ro tpm20-attestation-response* [tpm_name]
    |        +--ro tpm_name               string
    |        +--ro tpm-physical-index?    int32 {ietfhw:entity-mib}?
    |        +--ro up-time?               uint32
    |        +--ro node-name?             string
    |        +--ro node-physical-index?   int32 {ietfhw:entity-mib}?
    |        +--ro tpms-attest
    |        |  +--ro pcrdigest?                   binary
    |        |  +--ro tpms-attest-result?          binary
    |        |  +--ro tpms-attest-result-length?   uint32
    |        +--ro tpmt-signature?        binary
    +---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-name?                 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-name]
       |  |  +---w node-name                 string
       |  |  +---w node-physical-index?      int32 {ietfhw:entity-mib}?
       |  |  +---w (index-type)?
       |  |     +--:(last-entry)
       |  |     |  +---w last-entry-value?   binary
       |  |     +--:(index)
       |  |     |  +---w index-number?       uint64
       |  |     +--:(timestamp)
       |  |        +---w timestamp?          yang:date-and-time
       |  +---w log-type              identityref
       |  +---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-entry-quantity?   uint16
       +--ro output
          +--ro system-event-logs
             +--ro node-data* [node-name tpm_name]
                +--ro node-name              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 template-hash-algorithm?    string
                               +--ro template-hash?              binary
                               +--ro pcr-index?                  uint16
                               +--ro signature?                  binary
<CODE ENDS>
```

2.2. Raw Format

```
<CODE BEGINS>
module ietf-basic-remote-attestation {
  namespace "urn:ietf:params:xml:ns:yang:ietf-basic-remote-attestation";
  prefix "yang-brat";

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

  organization
    "Fraunhofer SIT";
  contact
    "Henk Birkholz
    Fraunhofer Institute for Secure Information Technology
    Email: henk.birkholz@sit.fraunhofer.de";
  description
    "A YANG module to enable TPM 1.2 and  TPM 2.0 based
    remote attestation procedures.
    Copyright (C) Fraunhofer SIT (2019).";
  revision "2019-07-08" {
```

```
    description
      "Second version";
    reference
      "draft-birkholz-rats-basic-yang-module";
    }

    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 TPM Algo ID.";
        }
      }
      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 TPM Algo ID.";
        }
      }
    }
  }

  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-name {
    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 of the 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 {
      description
        "In a distributed system with multiple compute nodes
         this is the node identified by name and physical-index.";
      leaf node-name {
         type string;
         description
           "Name of the compute node or All";
      }
      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";
    }
    leaf filename-hint {
      type string;
      description
        "File that was measured";
    }
    leaf filedata-hash {
      type binary;
      description
        "Hash of filedata";
    }
    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 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-name;
        }
```

```
        }
        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-name;
            uses node-uptime;
            uses compute-node;
            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 tpm20-pcr-selection;
```

```
        uses nonce;
        uses tpm20-signature-scheme;
        uses tpm20-attestation-key-identifier;
      }
      list tpms {
      key tpm_name;
      description
      "TPMs to fetch the attestation information.";
      uses tpm-name;
      }
    }
    output {
      list tpm20-attestation-response {
        key tpm_name;
        description
          "The binary output of TPM2b_Quote. An TPMS_ATTEST structure
          including a length, encapsulated in a signature";
        uses tpm-name;
        uses node-uptime;
        uses compute-node;
        container tpms-attest {
          leaf pcrdigest {
            type binary;
            description
              "split out value of TPMS_QUOTE_INFO for convenience";
          }
          leaf tpms-attest-result {
            type binary;
            description
              "The complete TPM generate structure including
              signature.";
          }
          leaf tpms-attest-result-length {
            type uint32;
            description
              "Length of attest result provided by the TPM structure.";
          }
          description
            "A composite of value and length and list of selected
            pcrs (original name: [type]attested)";
        }
        leaf tpmt-signature {
          type binary;
          description
            "Split out value of the signature for convenience.
            TODO: check for length values that complent binary value
            data node leafs.";
        }
```

```
      }
    }
  }

  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-name;
      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-name;
        uses node-uptime;
        uses compute-node;
        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-name;
          description
            "Selection of log entries to be reported.";
          uses compute-node;
          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 index-number {
                type uint64;
                description
                  "The 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.";
            }
          }
        }
```

```
        uses log-identifier;
        uses tpm20-pcr-selection;
        leaf log-entry-quantity {
          type uint16;
          description
            "The number of log entries to be returned. If omitted, it
            means all of them.";
        }
      }
      output {
        container system-event-logs {
          description
            "The requested data of the measurement event logs";
          list node-data {
            key "node-name tpm_name";
            description
              "Event logs of a node in a distributed system
               identified by the node name";
            uses compute-node;
            uses node-uptime;
            uses tpm-name;
            container log-result {
              description
                "The requested entries of the corresponding log.";
              uses event-logs;
            }
          }
        }
      }
    }

    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 tpms {
        key tpm_name;
        uses tpm-name;
        description
        "A list of TPMs in this composite
```

```
           device that rats can be conducted with.";
          list certificates {
            description
              "The TPM's endorsement-certificate.";
            container certificate {
              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 attestation-cert {
                    value 1;
                    description
                      "AK 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 assertions
                  (EK,AK Certificate). In a TPM 2.0 the EK,AK Certificate
                  resides in a well-defined NVRAM location by the TPM
                  vendor.";
              }
              description
                "Two kinds of certificates can be accessed via this
                statement. An Attestation Key Certificate and a
                Endorsement Key Certificate.";
            }
          }
        }
        list compute-nodes {
          key node-name;
          uses compute-node;
          description
            "A list names of hardware components in this composite
             device that rats can be conducted with.";
```

```
    }
  }
}
<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:

- o Addressed author's comments
- o Extended complementary details about attestation-certificates
- o Relabeled chunk-size to log-entry-quantity
- o Relabeled location with compute-node or tpm-name where appropriate
- o Added a valid entity-mib physical-index to compute-node and tpm-name to map it back to hardware inventory
- o Relabeled name to tpm_name
- o 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 Model for Challenge-Response-based Remote Attestation", draft-birkholz-rats-reference-interaction-model-00 (work in progress), March 2019.

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-10 (work in progress), July 2019.

[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>.

7.2. Informative References

[I-D.birkholz-attestation-terminology]
Birkholz, H., Wiseman, M., and H. Tschofenig, "Reference Terminology for Remote Attestation Procedures", draft-birkholz-attestation-terminology-02 (work in progress), July 2018.

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